

LNCS

# LNCS



Springer

# Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

196

---

## Advances in Cryptology:

Proceedings of CRYPTO 84

Edited by G.R. Blakley and David Chaum

---



Springer-Verlag  
Berlin Heidelberg New York Tokyo

## **Editorial Board**

D. Barstow W. Brauer P. Brinch Hansen D. Gries D. Luckham  
C. Moler A. Pnueli G. Seegmüller J. Stoer N. Wirth

## **Editors**

George Robert Blakley  
Department of Mathematics, Texas A&M University  
College Station, Texas 77843-3368, USA

David Chaum  
Center for Mathematics and Computer Science (CWI)  
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

CR Subject Classifications (1985): E.3

ISBN 3-540-15658-5 Springer-Verlag Berlin Heidelberg New York Tokyo  
ISBN 0-387-15658-5 Springer-Verlag New York Heidelberg Berlin Tokyo

Library of Congress Cataloging in Publication Data. CRYPTO 84: a Workshop on the Theory and Application of Cryptographic Techniques (1984: Santa Barbara, Calif.) CRYPTO 84. (Lecture notes in computer science; 196) Includes indexes. I. Computers—Access control—Congresses. 2. Cryptography—Congresses. I. Blakley, George Robert. II. Chaum, David. III. Title. IV. Series. QA76.9.A25C79 1984 005.8'2 85-14793  
ISBN 0-387-15658-5 (U.S.)

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks. Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to "Verwertungsgesellschaft Wort", Munich.

© by Springer-Verlag Berlin Heidelberg 1985  
Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.  
2145/3140-543210

## Preface

Here are some major contributions to the literature on modern cryptography: the papers presented at CRYPTO 84. It is our pleasure to share them with everyone interested in this exciting and growing field.

Each section of this volume corresponds to a session at the meeting. The papers were accepted by the program committee often only on the basis of abstracts, and appear here without having been otherwise refereed. The last section contains papers for some of the impromptu talks given at the traditional rump session. An author index as well as a keyword index, whose entries were mainly supplied by the authors, appear at the end of the volume.

The first two open meetings devoted to modern cryptography were organized independently: one by Allen Gersho during late Summer 1981 in Santa Barbara,<sup>1</sup> and the other by Thomas Beth and Rudiger Dierstein in Germany the following Spring.<sup>2</sup> David Chaum organized a successor to the Santa Barbara meeting the next year,<sup>3</sup> which launched the International Association for Cryptologic Research. The sponsorship of the association has continued the unbroken series of annual Summer CRYPTO meetings in the U.S.<sup>4</sup> and annual Spring EUROCRYPT meetings in Europe.<sup>5,6</sup>

It is our pleasure to thank all those who contributed to making these proceedings possible: the authors, program committee, other organizers of the meeting, IACR officers and directors, and all the attendees.

*College Station, Texas*  
*Amsterdam, the Netherlands*  
*March 1985*

G.R.B.  
D.C.

- 
1. Advances in Cryptology: A Report on CRYPTO 81, Allen Gersho, Ed., UCSB ECE Report no. 82-04, Department of Electrical and Computer Engineering, Santa Barbara CA 93106.
  2. Cryptography: Proceedings, Burg Feuerstein 1982, (Lecture Notes in Computer Science; 149) Thomas Beth Ed., Springer-Verlag, Berlin, 1983.
  3. Advances in Cryptology: Proceedings of CRYPTO 82, David Chaum, Ronald L. Rivest, and Alan T. Sherman, Eds., Plenum NY, 1983.
  4. Advances in Cryptology: Proceedings of CRYPTO 83, David Chaum Ed., Plenum NY, 1984.
  5. No proceedings were published for EUROCRYPT 83, which was held in Udine Italy.
  6. The proceedings of EUROCRYPT 84, Edited by Norbert Cot, are to appear in Lecture Notes in Computer Science, Springer-Verlag, Berlin.



# **CRYPTO 84**

*A Workshop on the Theory and Application of Cryptographic Techniques*

held at the University of California, Santa Barbara

August 19-22, 1984

sponsored by

*the International Association for Cryptologic Research*

## **Organizers**

Thomas A. Berson (Sytek, Inc.), General Chairman

G.R. Blakley (Texas A&M), Program Chairman

Henry Beker (Racal Research), Program

David Chaum (CWI), Proceedings

Dorothy Denning (SRI International), Program

Whitfield Diffie (BNR), Rump Session Chairman

Richard A. Kemmerer (UCSB), Local Arrangements

Ronald L. Rivest (MIT), Program

Miles Smidt (NBS), Program

Joe Tardo (DEC), Show & Tell

Kay G. White (Sytek, Inc.), Registration

# CONTENTS

## SECTION I: PUBLIC KEY CRYPTOSYSTEMS AND SIGNATURES

A Prototype Encryption System Using Public Key.....	3
<i>S.C. Serpell, C.B. Brookson, and B.L. Clark</i>	
A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.....	10
<i>Taher El Gamal</i>	
A Public Key Cryptosystem Based on the Word Problem.....	19
<i>Neal R. Wagner and Marianne R. Magyarik</i>	
Efficient Signature Schemes Based on Polynomial Equations.....	37
<i>H. Ong, C.P. Schnorr, and Adi Shamir</i>	
Identity-Based Cryptosystems and Signature Schemes.....	47
<i>Adi Shamir</i>	
A Knapsack Type Public Key Cryptosystem Based on Arithmetic in Finite Fields.....	54
<i>Benny Chor and Ronald L. Rivest</i>	
Some Public Key Crypto-Functions as Intractable as Factorization.....	66
<i>H.C. Williams</i>	

## SECTION II: CRYPTOSYSTEMS AND OTHER HARD PROBLEMS

Computing Logarithms in $GF(2^n)$ .....	73
<i>I.F. Blake, R.C. Mullin, and S.A. Vanstone</i>	
Wyner's Analog Encryption Scheme: Results of a Simulation.....	83
<i>Burt S. Kaliski</i>	
On Rotation Group and Encryption of Analog Signals.....	95
<i>Su-shing Chen</i>	
The History of Book Ciphers.....	101
<i>Albert C. Leighton and Stephen M. Matyas</i>	

An Update on Factorization at Sandia National Laboratories .....	114
<i>J.A. Davis and D.B. Holdridge</i>	
An LSI Digital Encryption Processor (DEP) .....	115
<i>R.C. Fairfield, A. Matusевич, and J. Planý</i>	
Efficient Hardware and Software Implementations for the DES .....	144
<i>Marc Davio, Yvo Desmedt, Jo Goubert, Frank Hoornaert, and Jean-Jacques Quisquater</i>	
Efficient Hardware Implementation of the DES .....	147
<i>Frank Hoornaert, Jo Goubert, and Yvo Desmedt</i>	
A Self-Synchronizing Cascaded Cipher System with Dynamic Control of Error Propagation .....	174
<i>Norman Proctor</i>	

### SECTION III: RANDOMNESS AND ITS CONCOMITANTS

Efficient and Secure Pseudo-Random Number Generation .....	193
<i>Umesh V. Vazirani and Vijay V. Vazirani</i>	
An LSI Random Number Generator (RNG) .....	203
<i>R.C. Fairfield, R.L. Mortenson, and K.B. Coulthart</i>	
Generalized Linear Threshold Scheme .....	231
<i>S.C. Kothari</i>	
Security of Ramp Schemes .....	242
<i>G.R. Blakley and Catherine Meadows</i>	
A Fast Pseudo Random Permutation Generator with Applications to Cryptology .....	269
<i>Selim G. Akl and Henk Meijer</i>	
On the Cryptographic Applications of Random Functions .....	276
<i>Oded Goldreich, Shafi Goldwasser, and Silvio Micali</i>	
An Efficient Probabilistic Public Key Encryption Scheme which Hides All Partial Information .....	289
<i>Manuel Blum and Shafi Goldwasser</i>	

### SECTION IV: ANALYSIS AND CRYPTANALYSIS

RSA/Rabin Least Significant Bits are $\frac{1}{2} + 1 / \text{poly}(\log N)$ Secure .....	303
<i>Benny Chor and Oded Goldreich</i>	
Information Theory without the Finiteness Assumption, I: Cryptosystems as Group-Theoretic Objects .....	314
<i>G.R. Blakley</i>	
Cryptanalysis of ADFGVX Encipherment Systems .....	339
<i>Alan G. Konheim</i>	

Breaking Iterated Knapsacks .....	342
<i>Ernest F. Brickell</i>	
Dependence of Output on Input in DES: Small Avalanche Characteristics.....	359
<i>Yvo Desmedt, Jean-Jacques Quisquater, and Marc Davio</i>	
DES Has No Per Round Linear Factors.....	377
<i>J.A. Reeds and J.L. Manferdelli</i>	

## SECTION V: PROTOCOLS AND AUTHENTICATION

A Message Authenticator Algorithm Suitable for a Mainframe Computer.....	393
<i>Donald Watts Davies</i>	
Key Management for Secure Electronic Funds Transfer in a Retail Environment.....	401
<i>Henry Beker and Michael Walker</i>	
Authentication Theory/Coding Theory .....	411
<i>Gustavus J. Simmons</i>	
New Secret Codes Can Prevent a Computerized Big Brother.....	432
<i>David Chaum</i>	
Fair Exchange of Secrets .....	434
<i>Tom Tedrick</i>	
Cryptoprotocols: Subscription to a Public Key, the Secret Blocking and the Multi-Player Mental Poker Game.....	439
<i>Mordechai Yung</i>	
Poker Protocols.....	454
<i>Steven Fortune and Michael Merritt</i>	

## SECTION VI: IMPROMPTU TALKS

A 'Paradoxical' Solution to the Signature Problem.....	467
<i>Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest</i>	
Sequence Complexity as a Test for Cryptographic Systems.....	468
<i>A.K. Leung and S.E. Tavares</i>	
An Update on Quantum Cryptography.....	475
<i>Charles H. Bennett and Gilles Brassard</i>	
How to Keep a Secret Alive: Extensible Partial Key, Key Safeguarding, and Threshold Systems .....	481
<i>David Chaum</i>	
Author Index.....	487
Keyword Index .....	489

## A PROTOTYPE ENCRYPTION SYSTEM USING PUBLIC KEY

Authors: S C Serpell, C B Brookson and B L Clark.

British Telecom Research Laboratories, Martlesham Heath, Ipswich, Suffolk, IP5 7RE. United Kingdom.

### BACKGROUND

The use of cryptography to produce a secure method of user authentication and to encipher traffic on data or digital links has been the aim of many of those defining theoretical schemes and techniques. This paper describes an experimental realisation of these aims in hardware, in order to provide a secure and authenticated communications channel.

The communications channel in this case could be part of almost any service, such as videotex, teletex, Local Area Networks, packet data systems, telex or conventional telephone data links.

### SYSTEM REQUIREMENTS

The basic objective is that authorised users, and only authorised users, should be able to exchange meaningful data over a link. This statement actually conceals not one but two tasks; firstly, verifying the identities of the would-be link users to one another (authentication), and secondly, processing their data so as to make it unintelligible to eavesdroppers (encipherment). Both criteria have to be met by electronic means without any excessive operational constraints or overheads.

The traditional approach to the problem is for users' encryption equipments to deploy a conventional algorithm such as British Telecom's B-Crypt (1) or the American Data Encryption Standard DES (2). Users exchanging information have to be party to the same key. Successful interworking then gives implicit proof of the user identities, but the need for the secure distribution of pre-agreed secret keys represents a massive overhead which becomes impractical in large systems. It is impossible to set up a secure communications link without an exchange of keys prior to link set-up, and this represents a severe if not prohibitive drawback in services such as LANs or teletex.

Another approach is to use a public key algorithm such as PSA (3). This can reduce the key management overhead to an acceptable level and at the same time authenticate user identity, allowing the secure links to be established between strangers. But most implementations of public key so far have been relatively inefficient, generally limited to a speed of operation of a few tens of bits per second. Thus they do not support the encipherment rates needed in many applications (4).

However, by combining the conventional and public key methods in a system, it is possible to obtain the best of both worlds. Public key procedures are used at link set up time to prove user identities and establish a secret key which is used in the conventional encipherment of the data. This combination, in order to be applicable to most communications links and easy to use, needs to be defined carefully to overcome any inherent risks and to obtain an operationally secure system.

The experimental system described here was designed to meet the

following requirements:

SECURITY To use public keys of up to 512 bits and any conventional algorithm desired. No long-term sensitive information to be within the equipment.

SPEED The equipment should allow automatic set up of a secure link within 3 seconds.

COMMUNICATIONS LINK The units to be able to work on most types of links, at transmission speeds of up to 19 bytes, and be tolerant of link errors.

USER FRIENDLINESS The interface and requirements placed on the system user to be secure as possible.

USER IDENTITY The individual user to be identified by tokens, defining him and his access rights and privileges. The token needs to be unforgeable.

USERS It should be possible to use the equipment for inter-terminal communications as well as for services involving hosts and databases.

#### ENCIPHERING METHOD

##### Bilateral Communications

A participant A in an RSA-type public key system is provided with three entities; an exponent EA and a modulus MA which are public knowledge and comprise the 'public key', and a further exponent DA which is known to A alone and comprises the 'secret key'. A message P may be enciphered by raising P to the power EA (or DA) modulo MA to form ciphertext C. The original plaintext P is recovered by raising C to the power DA (or EA) modulo MA. Since DA is known only to A, this permits the following authentications:

- Only A can recover the message enciphered under EA, so a user who enciphers data under A's public key can be confident that only A may understand the message,
- Only A could originate a message enciphered under DA, so a user who decipheres a message using A's public key can be confident that only A could have sent the message.

A number of possible public key protocols are possible which take advantage of these properties (5,6). The set-up protocol adopted in the experimental system create a random number R between A and B, keeping R secure against eavesdropping, as follows:

- 1 An unenciphered link is established.
- 2 A invents a random number RA, and enciphers it under B's public key to form  $SA = EA \exp (EB) \pmod{MB}$ . Similarly, B forms  $SB = RB \exp (EA) \pmod{MA}$ .
- 3 The users exchange SA, SB.
- 4 User A recovers  $RB - SB \exp (DA) \pmod{MA}$ , user B recovers  $RA = SA \exp (DB) \pmod{MB}$ .
- 5 Both users form  $R = RA \text{ xor } RB$ .

The conventional key subsystem then derives a key K so that the users may communicate:

- 6 Both users truncate R in the same way to obtain a key K and a initialisation vector IV.

7 Both users load K to encrypt all following data using the conventional algorithm.

Unless both parties use the same K and commence secure communication with the same initialisation vector IV, the information exchange will fail, so authentication is complete in the first subsequent exchange of data.

### Multiple Users

The simple protocol above can be modified to include multiple users, so that:

- 1 User A creates  $RA$  and forms  $SAB = RA \exp(EB) \pmod{MB}$ ,  $SAC = RA \exp(EC) \pmod{MC}$ ...., while user B forms  $SBA$ ,  $SBC$ ...., C forms  $SCA$ ,  $SCB$ .... etc.
- 2 Users then exchange  $SAB$ ,  $SAC$ ....,  $SBA$ ,  $SBC$ ....,  $SCA$ ,  $SCB$ .... etc.
- 3 A then recovers  $RB = SBA \exp(DA) \pmod{MA}$ ,  $RC = SCA \exp(DA) \pmod{MA}$ ...., B recovers  $RA$ ,  $RC$ ...., and C recovers  $RA$ ,  $RB$ .... etc.

The conventional algorithm is then used for communication as before.

### Certification of Keys and Privileges

In a secure system a user could obtain the public key and privileges of access of his desired partner by interrogating a reliable public key directory. This limits the flexibility and response time of the system as a secure call to the directory must be made before key exchange can take place.

A method of certifying keys is therefore introduced using a system public key known to an issuing authority. Extra steps are needed to establish the certificates which are then issued to the users:

- a System public key values  $ES$ ,  $MS$  and  $DS$  are established.
- b  $ES$ ,  $MS$  are widely distributed so that each user is assured of their values;  $DS$  is kept secret by the issuing authority.
- c each user is issued with a certificate of his identity, public key, and any other useful information (eg. access rights, privileges, name etc) protected using the system secret key  $DS$ . Thus user A's certificate could be  $CA = ZA \exp(DS) \pmod{MS}$  where  $ZA$  is the string formed by the concatenation of  $A:EA:MA:....$ , or alternatively a certificate could be formed by appending a banking- message type authenticator at the end of the string  $A:EA:MA....$  based on  $DS$  (7).

The public key protocol now commences:

- 1 Users A, B establish unenciphered links.
- 1a Users A, B exchange certificates.
- 1b User A obtains B's public key  $EB$ ,  $MB$  from  $ZB = CB \exp(ES) \pmod{MS}$ , while B similarly obtains A's public key.

This certificate system is also available for multiple users.

### SYSTEM REALISATION

The experimental hardware is built to realise the protocols and enciphering processes already described. The ease of use of the hardware, together with the security it could afford, were the two maxims that were employed when designing the system. Figure 1 shows the system

block diagram.

### Equipment Interface

The system is housed in a stand-alone case, and uses V24 (RS232) serial interfaces to interconnect the user (terminal or system) to the communications link (modems, local area networks etc). Another replica unit is employed by the communicant at the other end of the communications link.

### Token

The certified part of the key, ZA for user A (containing his privileges and public key), his secret key (DA) and A's copy of the system public key ES and MS are stored in this realisation on a DataKey (8), which is essentially a block of memory in the form of some Electrically Alterable Read Only Memory (EAROM) encased in plastic in the shape of a key. This key is certified, and the secret key is protected by being combined with an seven digit randomly selected number which is the Personal Identity Number (PIN) of the key holder. Other forms of token, such as magnetic cards, are equally suitable.

### Use

The link is first established between the users in unencrypted mode, and the user requiring encipherment inserts his key into a receptacle. The key is then read, and the user is requested to enter his PIN by a message displayed on a liquid crystal panel. After his PIN has been entered, a message is sent to the distant user inviting him to insert his key and enter his PIN. The exchange protocol then takes place, and each user should then finish up with the same random number, which is used to derive the conventional enciphering key and initialisation vector.

Finally, a verification message is then passed between enciphering systems to establish that the exchange has been successful. Each user is able to inspect the certified names of his partners as they are displayed on his own enciphering equipment. The user equipment is now able to conduct an enciphered communications session.

The link is terminated by the withdrawal of a key, and the enciphering units are returned to plaintext mode.

### HARDWARE DEVELOPED

#### User View

The hardware developed in the initial phase consists of the user interface presented to the operator and three other sub-systems described below which implement the protocols adopted. The sub-systems are integrated into a small rack unit in this prototype realisation.

The user connects the enciphering unit between his terminal and the communications link using the serial ports. He is presented with four interfaces:

- A display to provide plain language instructions, status and fault conditions,
- An audio sounder to draw attention to operating conditions requiring attention,



- A numeric keyboard to input the PIN,
- A key receptacle for the DataKey.

### Exponentiator

A hardware public key exponentiator capable of exponentiation of numbers up to 512 bits in length. This uses conventional TTL integrated circuits. The exponentiation of a 512 bit number takes about one second. The exponentiator connects to a controlling microprocessor by serial interfaces.

### Stream Cipher

The conventional encryption is performed by equipment using B-Crypt in additive stream cipher mode or DES in cipher feedback mode. The control microprocessor loads the key and IV through a specially dedicated serial port, and the plaintext and ciphertext through a pair of serial ports.

### Control Microprocessor

An Intel 8085 is used as the control microprocessor. Serial ports connect the control circuitry to the user and communications link, and the exponentiator and stream encryptor. Parallel ports control liquid crystal display and PIN input pad. The protocol is implemented in software.

### RESULTS

Two experimental units have been produced, and these have been tested on various links such as on a local area network and a modem line connection. The hardware has operated reliably, and is capable of going into encrypted mode within 3 seconds of initiation. This interval of time has proved acceptable to users.

The experimental equipment produced has proved to be a valuable step towards the proof of the system viability. This concept forms the basis of a 'universal' encryption system, and further continuing evaluation and development is being carried out. The particular conventional and public key algorithms used in the initial stage are also subject to further scrutiny and evaluation. The overall system size is being reduced by adopting integration techniques on the various circuit elements.

### ACKNOWLEDGEMENTS

Acknowledgement is made to the Director of Research of British Telecom for permission to publish this paper.

### REFERENCES

1. B-Crypt Specification, BT Cryptographic Products, June 1984.
2. 'Data Encryption Standard', National Bureau of Standards, FIPS PUB 46, 1977.
3. Rivest, R L, Shamir, A, and Adleman, L, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Comm. ACM, 1978, 21, Pp 120-126.
4. Davies, D W, Price, W L, and Parkin, G I, "Evaluation of Public Key Cryptosystems", Information Privacy, 1980, 2, Pp 138-154.
5. Schanning, B P, "Applying Public Key Distribution to Local Area Networks", Computers and Security, 1982, 1, Pp 268-274.

6. Ingemarsson, I, "A Conference Key Distribution System", IEEE Trans, Inf. Theory, 1982, IT-28, Pp 714-720.
7. American National Standard X9.9, "Financial Institution Message Authentication".
8. 'DataKey' Handbook, DataCard International, drayton House, Chichester, Sussex, March 1983.

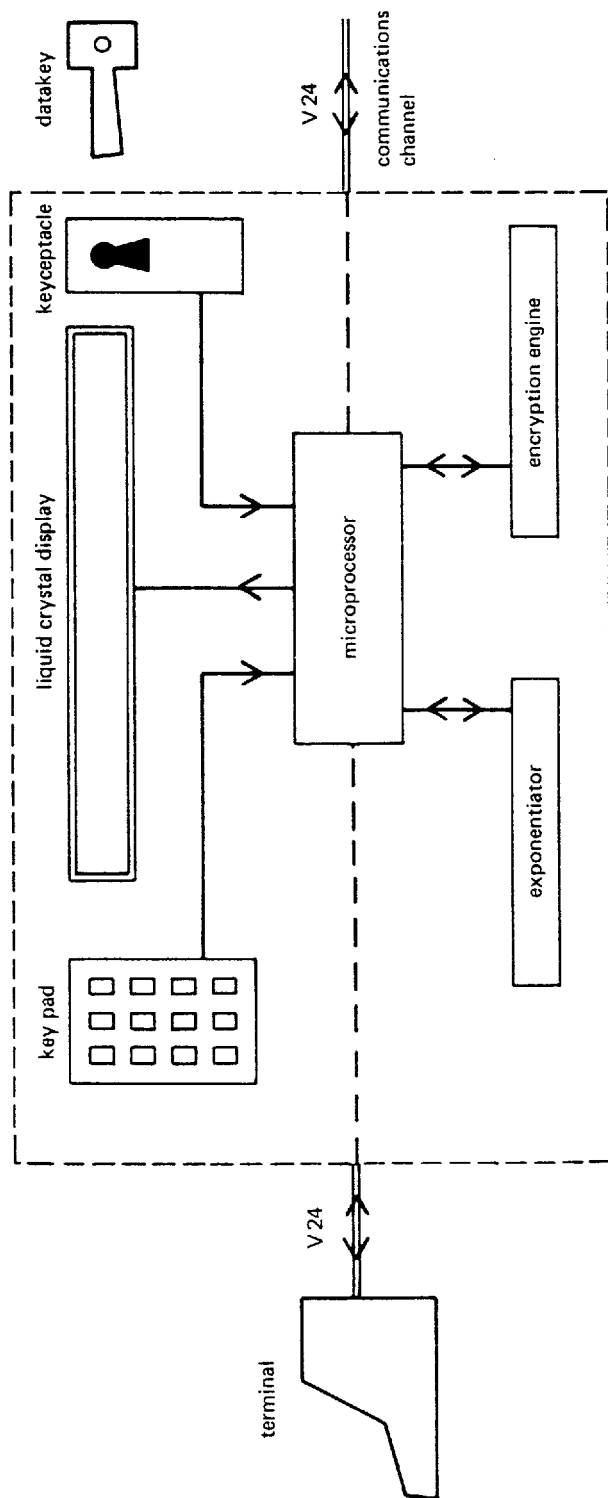


Figure 1 Public key cryptosystem realisation

# A PUBLIC KEY CRYPTOSYSTEM AND A SIGNATURE SCHEME BASED ON DISCRETE LOGARITHMS

TaherElGamal\*

Hewlett-Packard Labs  
1501 Page Mill Rd  
Palo Alto CA 94301

## ABSTRACT

A new signature scheme is proposed together with an implementation of the Diffie - Hellman key distribution scheme that achieves a public key cryptosystem. The security of both systems relies on the difficulty of computing discrete logarithms over finite fields.

## 1. INTRODUCTION

In 1976, Diffie and Hellman [3] introduced the concept of public key cryptography. Since then, several attempts have been made to find practical public key systems (see for example [8,7,9]) depending on the difficulty of solving some problems. For example, the RSA system [9] depends on the difficulty of factoring large integers. This paper presents systems that rely on the difficulty of computing logarithms over finite fields.

Section 2 shows a way to implement the public key distribution scheme introduced by Diffie and Hellman [3] to encrypt and decrypt messages. The security of this system is equivalent to that of the distribution scheme. Section 3 introduces a new digital signature scheme that depends on the difficulty of computing discrete logarithms over finite fields. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. Section 4 develops some attacks on the signature scheme, none of which seems to break it.

---

This work was supported by the NSF under contract ECS83 07741 while the author was at the information systems laboratory, Stanford University.

Section 5 gives some properties of the system. Section 6 concludes and gives some remarks.

## 2. THE PUBLIC KEY SYSTEM

First, the Diffie - Hellman Key distribution scheme is reviewed. Suppose that A and B want to share a secret  $K_{AB}$  where A has a secret  $x_A$ , and B has a secret  $x_B$ . Let  $p$  be a large prime and  $\alpha$  be a primitive element mod  $p$ , both known. A computes  $y_A = \alpha^{x_A} \bmod p$ , and sends  $y_A$ . Similarly B computes  $y_B = \alpha^{x_B} \bmod p$  and sends  $y_B$ . Then the secret  $K_{AB}$  is computed as

$$\begin{aligned} K_{AB} &= \alpha^{x_A x_B} \bmod p \\ &= y_A^{x_B} \bmod p \\ &= y_B^{x_A} \bmod p. \end{aligned}$$

Hence both A and B are able to compute  $K_{AB}$ . But for an intruder computing  $K_{AB}$  appears to be difficult. Note that it is not yet proved that breaking the system is equivalent to computing discrete logarithms. For more details refer to [3].

In any of the cryptographic systems that are based on discrete logarithms,  $p$  must be chosen such that  $p - 1$  has at least one large prime factor. If  $p - 1$  has only small prime factors, then computing discrete logarithms is easy (see [8]).

Now suppose that A wants to send B a message  $m$ , where  $0 \leq m \leq p - 1$ . First A chooses a number  $k$  uniformly between 0 and  $p - 1$ . Note that  $k$  will serve as the secret  $x_A$  in the key distribution scheme. Then A computes the "key"

$$K = y_B^k \bmod p, \quad (1)$$

where  $y_B = \alpha^{x_B} \bmod p$  is either in a public file, or is sent by B. The encrypted message (or ciphertext) is then the pair  $(c_1, c_2)$ , where

$$c_1 = \alpha^k \bmod p, \quad c_2 = K m \bmod p, \quad (2)$$

and  $K$  is computed in (1).

Note that the size of the ciphertext is double the size of the message. Also note that the multiplication operation in (2) can be replaced by any other invertible operation such as addition mod  $p$ .

The decryption operation splits into 2 parts. The first step is recovering  $K$ , which is easy for B since  $K = (\alpha^k)^{x_B} = c_1^{x_B} \bmod p$ , and  $x_B$  is known to B only. The second step is to divide  $c_2$  by  $K$  and recover the message  $m$ .

The public file consists of one entry for each user, namely  $y_i$  for user  $i$  (since  $\alpha$  and  $p$  are known for all users). It is possible that each user chooses his own  $\alpha$  and  $p$ , which is preferable

from the security point of view but that will triple the size of the public file.

It is not advisable to use the same value  $k$  for enciphering more than one block of the message, since if  $k$  is used more than once, knowledge of one block  $m_1$  of the message enables an intruder to compute other blocks as follows:

Let

$$c_{1,1} = \alpha^k \bmod p, \quad c_{2,1} = m_1 K \bmod p,$$

and

$$c_{1,2} = \alpha^k \bmod p, \quad c_{2,2} = m_2 K \bmod p.$$

Then  $\frac{m_1}{m_2} = \frac{c_{2,1}}{c_{2,2}} \bmod p$ , and  $m_2$  is easily computed if  $m_1$  is known.

It can be easily seen that breaking the system is equivalent to breaking the Diffie - Hellman distribution scheme. First, if  $m$  can be computed from  $c_1$ ,  $c_2$ , and  $y$ , then  $K$  can also be computed from  $y$ ,  $c_1$ , and  $c_2$  (which appears like a random number since  $k$  and  $m$  are unknown). That is equivalent to breaking the distribution scheme. Second, (even if  $m$  is known) computing  $k$  or  $x$  from  $c_1$ ,  $c_2$ , and  $y$  is equivalent to computing discrete logarithms. The reason is that both  $x$  and  $k$  appear in the exponent in  $y$  and  $c_1$ .

### 3. A DIGITAL SIGNATURE SCHEME

A new signature scheme is described in this section. The public file contains the same public keys for encrypting messages as well as verifying signatures.

Let  $m$  be a document to be signed, where  $0 \leq m \leq p - 1$ . The public file still consists of the public key  $y = \alpha^x \bmod p$  for each user. To sign a document, a user  $A$  should be able to use the secret key  $x_A$  to find a signature for  $m$  in such a way that all users can verify the authenticity of the signature using the public key  $y_A$  (together with  $\alpha$  and  $p$ ), and no one can forge a signature without knowing the secret  $x_A$ .

The signature for  $m$  is the pair  $(r, s)$ ,  $0 \leq r, s < p - 1$ , chosen such that the equation

$$\alpha^m = y^r r^s \bmod p \tag{3}$$

is satisfied.

#### 3.1. The Signing Procedure

The signing procedure consists of the following 3 steps:

A. Choose a random number  $k$ , uniformly between 0 and  $p - 1$ , such that

$$\gcd(k, p - 1) = 1.$$

B. Compute

$$r = \alpha^k \bmod p. \quad (4)$$

C. Now (3) can be written as

$$\alpha^m = \alpha^{x r} \alpha^{k s} \bmod p, \quad (5)$$

which can be solved for  $s$  using

$$m = x r + k s \bmod (p - 1). \quad (6)$$

Equation (6) has a solution for  $s$  if  $k$  is chosen such that  $\gcd(k, p - 1) = 1$ .

### 3.2. The Verification Procedure

Given  $m$ ,  $r$ , and  $s$ , it is easy to verify the authenticity of the signature by computing both sides of (3) and checking that they are equal.

Note

As will be shown in section 4, the value of  $k$  chosen in step A should never be used more than once. This can be guaranteed, for example, by using as a " $k$  generator" a DES chip used in the counter mode as a stream cipher.

## 4. SOME ATTACKS ON THE SIGNATURE SCHEME

This section introduces some of the possible attacks on the signature scheme. Some of these attacks are easily shown to be equivalent to computing discrete logarithms over  $GF(p)$ . It is not yet proved that breaking the signature scheme is equivalent to computing discrete logarithms, or equivalent to breaking the distribution scheme. However, none of the attacks shown in this section appear to break the system. The reader is encouraged to develop new attacks, or find fast algorithms to perform one of the attacks described in this section. The attacks will be divided into two groups. The first group includes some attacks for recovering the secret key  $x$ , and in the second group we show some attacks for forging signatures without recovering  $x$ .

### 4.1. Attacks aiming to recover $x$

4.1.1. Given  $\{m_i : i = 1, 2, \dots, l\}$  documents, together with the corresponding signatures  $\{(r_i, s_i) : i = 1, 2, \dots, l\}$ , an intruder may try to solve  $l$  equations of the form (6). Since there are  $l + 1$  unknowns (since each signature uses a different  $k$ ), the system of

equations is underdetermined and the number of solutions is large. The reason is that each value for  $x$  yields a solution for the  $k_i$ 's since a system of linear equations with a diagonal matrix of coefficients will result. Since  $p - 1$  is chosen to have at least one large prime factor  $q$ , recovering  $x \bmod q$  requires an exponential number of message-signature pairs.

#### Note

If any  $k$  is used twice in the signing, then the system of equations is uniquely determined and  $x$  can be recovered. So for the system to be secure, any value of  $k$  should never be used twice.

4.1.2. Trying to solve equations of the form (3) is always equivalent to computing discrete logarithms over  $GF(p)$ , since both unknowns  $x$ , and  $k$  appear in the exponent.

4.1.3. An intruder might try to develop some linear dependencies among the unknowns  $\{k_i, i = 1, 2, \dots, l\}$ . This is also equivalent to computing discrete logarithms since if  $k_i = c k_j \bmod (p - 1)$ , then  $r_i = r_j^c \bmod p$ , and if  $c$  can be computed then computing discrete logarithms is easy.

## 4.2. Attacks for Forging Signatures

4.2.1. Given a document  $m$ , a forger may try to find  $r, s$  such that (3) is satisfied. If  $r = \alpha^j \bmod p$  is fixed for some  $j$  chosen at random, then computing  $s$  is equivalent to solving a discrete logarithm problem over  $GF(p)$ .

If the forger fixes  $s$  first then  $r$  could be computed from the equation

$$r^s y^r = A \bmod p. \quad (7)$$

Solving equation (7) for  $r$  is not yet, proved to be at least as hard as computing discrete logarithms, but we believe that it is not feasible to solve (7) in polynomial time. The reader is encouraged to find a polynomial time algorithm for solving (7).

4.2.2. It seems possible that (3) can be solved for both  $r$  and  $s$  simultaneously, but we have not been able to find an efficient algorithm to do that.

4.2.3. The signature scheme allows the following attack, whereby the intruder, knowing one legitimate signature for one message, can generate other legitimate signatures and messages. This attack does not allow the intruder to sign an arbitrary message and therefore does not break the system. This property exists in all the existing digital signature schemes and can be avoided by either requiring that  $m$  has to be of certain structure, or by applying a



one-way function to the message  $m$  before signing it.

Given a signature  $(r, s)$  for the message  $(m)$ , then

$$\alpha^m = y^r r^s \bmod p.$$

Select integers  $A$ ,  $B$ , and  $C$  arbitrarily, such that  $(Ar - Cs)$  is relatively prime to  $p - 1$ . Set

$$r' = r^A \alpha^B y^C \bmod p,$$

$$s' = sr' / (Ar - Cs) \bmod (p - 1),$$

$$\text{and } m' = r'(Am + Bs) / (Ar - Cs) \bmod (p - 1).$$

Then it is claimed that  $(r', s')$  signs the message  $(m')$ . Calculate

$$\begin{aligned} y^{r'} r'^{s'} &= y^{r'} (r^A \alpha^B y^C)^{sr' / (Ar - Cs)} \\ &= (y^{r' Ar - r' Cs} + r' Cs \cdot r^{Asr'} \alpha^{Bsr'})^{1 / (Ar - Cs)} \\ &= ((y^r r^s)^{Ar - Cs} \alpha^{Bsr'})^{1 / (Ar - Cs)} \\ &= \alpha^{(mAr + Bs) / (Ar - Cs)} \\ &= \alpha^{m'} \quad (\text{all calculations } \bmod p). \end{aligned}$$

As a special case, setting  $A = 0$ , legitimate signatures can be generated with corresponding messages without ever seeing any signatures:

$$r' = \alpha^B y^C \bmod p,$$

$$s' = -r' / C \bmod (p - 1),$$

$$m' = -r' B / C \bmod (p - 1).$$

It can be shown that  $(r', s')$  signs  $(m')$ .

## 5. PROPERTIES OF OUR SYSTEM AND COMPARISON TO OTHER SIGNATURE SCHEMES AND PUBLIC KEY SYSTEMS

Let  $m$  be the number of bits in either  $p$  for the discrete logarithm problem, or  $n$  for the integer factoring problem. Then the best known algorithm for both computing discrete logarithms and factoring integers (which is the function used in some of the existing systems such as the RSA system [9]) is given by (see [1,5,10])

$$O(\exp \sqrt{c m \ln m}), \quad (8)$$

where the best estimate for  $c$  is  $c = 0.69$  for factoring integers, (due to Schnorr and Lenstra [10]), as well as for discrete logarithms over  $GF(p)$  (see [5]). These estimates imply that we have to use numbers of about the size of the numbers used in the RSA system to obtain the same level of security (assuming the current value for  $c$  for both the discrete logarithms problem and the integer factorization problem). So, the size of the public file is larger than that for the RSA system. (For the RSA system, each user has one entry  $n$  as his public key together with the encryption key in the public file.)

### 5.1. Properties of the public key system

As shown above our system has some differences with the other known systems. First, due to the randomization in the enciphering operation, the cipher text for a given message  $m$  is not repeated, i.e. if we encipher the same message twice we will not get the same cipher text  $\{c_1, c_2\}$ . This prevents attacks like a probable text attack where if the intruder suspects that the plain text is, for example,  $m$ , then he tries to encipher  $m$  and finds out if it was really  $m$ . This attack, and similar ones, will not succeed since the original sender chose a random number  $k$  for enciphering, and different values of  $k$  will yield different values of  $\{c_1, c_2\}$ . Also, due to the structure of our system, there is no obvious relation between the enciphering of  $m_1$ ,  $m_2$ , and  $m_1 m_2$ , or any other simple function of  $m_1$  and  $m_2$ . This is not the case for the known systems, such as the RSA system.

Suppose that  $p$  is of about the same size as that required for  $n$  in the case of the RSA system. Then the size of the cipher text is double the size of the corresponding RSA cipher text.

For the enciphering operation two exponentiations are required. That is equivalent to about  $2 \log p$  multiplications in  $GF(p)$ . For the deciphering operation only one exponentiation (plus one division) is needed.

### 5.2. Properties of the signature scheme

For the signature scheme using the above arguments for the sizes of the numbers in our system and the RSA system, the signature is double the size of the document. Then, the size of the signature is of the same size as that needed for the RSA scheme, and half the size of the signature for the new signature scheme that depends on quadratic forms published by Ong and Schnorr[8], and also Ong, Schnorr, and Shamir[7] (since both systems are based on the integer factoring problem). The Ong-Schnorr-Shamir system has been broken by Pollard and new variations are being suggested. Thus it is not clear at the present time whether a secure system based on modular equations can be found and hence no further remarks will be made regarding these schemes.

Note that, since the number of signatures is  $p^2$  while the number of documents is only  $p$ , that each document  $m$  has a lot of signatures, but any signature signs only one document.

For the signing procedure, one exponentiation (plus a few multiplications) is needed. To verify a signature, it seems that three exponentiations are needed, but it was pointed to the author by A. Shamir that only 1.875 exponentiations are needed. This is done by representing

the three exponents  $m$ ,  $r$ ,  $s$  in their binary expansion. At each step square the number  $\alpha^{-1}yr$  and divide by the necessary factor to account for the different expansions of  $m$ ,  $r$ , and  $s$ . The different multiples of  $\alpha^{-1}$ ,  $y$ , and  $r$  can be stored in a table consisting of eight entries. We expect that 0.875 of the time a multiplication is needed. That accounts for the 1.875 exponentiations needed.

## 6. CONCLUSIONS AND REMARKS

The paper described a public key cryptosystem and a signature scheme based on the difficulty of computing discrete logarithms over finite fields. The systems are only described in  $GF(p)$ . The public key system can be easily extended to any  $GF(p^m)$ , but due to recent progress in computing discrete logarithms over  $GF(p^m)$ , where  $m$  is large (see [2,5]), it is advisable to use  $GF(p)$  instead since it seems that it is harder to compute logarithms over  $GF(p)$  than over  $GF(q^m)$  for large  $m$  if  $p$  and  $q^m$  are of the same size. The subexponential time algorithm has been extended to  $GF(p^2)$  [4] and it appears that it can be extended to all finite fields. Hence, it seems that it is better to use  $GF(p)$  for implementing any cryptographic system. The estimates for the running time of computing discrete logarithms and factoring integers are the best known so far. These estimates imply that the public file size is larger in this scheme than in the RSA scheme, but the difference is at most a factor of two due to the structure of both schemes. Also the size of the cipher text is double that of the RSA system.

## ACKNOWLEDGEMENT

The author would like to thank a referee for including the attack described in section 4.2.3.

## REFERENCES

- [1] L. Adleman, A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography, Proc. 20th IEEE Foundations of Computer Science Symposium (1979), 55-80.
- [2] D. Coppersmith, Fast Evaluation of Logarithms in Fields of Characteristic two, IEEE Transactions on Information Theory IT-30 (1984), 587-594.

- [3] W. Diffie and M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, IT-22 (1976), 472-492.
- [4] T. ElGamal, A Subexponential-Time Algorithm for Computing Discrete Logarithms over  $GF(p^2)$ , submitted to IEEE Transactions on Information Theory.
- [5] A. Odlyzko, Discrete Logarithms in Finite Fields and Their Cryptographic Significance, to appear in Proceedings of Eurocrypt '84.
- [6] H. Ong and C. Schnorr, Signatures Through Approximate Representations by Quadratic Forms, to be published.
- [7] H. Ong, C. Schnorr, and A. Shamir, An efficient Signature Scheme Based on Quadratic Forms, pp 208-216 in Proceedings of 16th ACM Symposium on Theoretical Computer Science, 1984.
- [8] S. Pohlig and M. Hellman, An improved algorithm for computing logarithms over  $GF(p)$  and its cryptographic significance, IEEE Transactions on Information Theory It-24 (1978), 106-110.
- [9] R. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of the ACM, Feb 1978, volume 21, number 2, 120-126.
- [10] C. Schnorr and H. W. Lenstra Jr., A Monte Carlo Factoring Algorithm with Finite Storage, to appear in Mathematics of Computation.

# A PUBLIC-KEY CRYPTOSYSTEM BASED ON THE WORD PROBLEM

Neal R. Wagner<sup>1</sup>  
Marianne R. Magyarik

Drexel University  
Mathematics and Computer Science  
Philadelphia, Pennsylvania 19104

## ABSTRACT.

The undecidable word problem for groups and semigroups is investigated as a basis for a public-key cryptosystem. A specific approach is discussed along with the results of an experimental implementation. This approach does not give a provably secure or practical system, but shows the type of cryptosystem that could be constructed around the word problem. This cryptosystem is randomized, with infinitely many ciphertexts corresponding to each plaintext.

## 1. NP-COMPLETE PROBLEMS.

The idea of using an NP-complete problem to construct a public-key cryptosystem (PKC) seemed promising [Diff76], but has not been successful historically. The earliest such PKC was based on the integer knapsack problem, and recently various versions of this PKC have been broken by general, powerful attacks [Sha83a], [Adle83]. (In this case, the attacks have been carried out on the type of trapdoor inserted, and not directly on the

---

<sup>1</sup> This work was supported in part by NSF grant DCR-8403350, and by Drexel University's Faculty Development Mini-Grant program

knapsack problem itself.) Other PKC's based on NP-complete problems have been proposed, but none seems successful so far.

There is a hierarchy of decision problems, from simplest to hardest, extending from polynomial-time problems, through NP-complete problems, to the undecidable problems (the hardest of all) [Tarj83], [Aho74].

NP-complete problems are often regarded as lying on the "boundary" of intractability, i.e., they are the simplest known "natural" problems which are intractable. Stated differently, if an NP-complete problem is even slightly weakened, it may no longer be intractable. In constructing a PKC, one must insert a trapdoor, and after allowing for all the various kinds of attacks, we would not usually expect to have a "pure" NP-complete problem remaining.

Along similar lines Brassard [Bras79] has shown that, with a few restrictions, if a cryptosystem were provably NP-complete to break, then the theoretical result NP= co-NP would follow. This latter result is widely conjectured to be untrue, though no proof is available [Gary79]. Thus the cryptanalysis of a PKC based on an NP-complete problem would be easier than NP-complete, hence likely a tractable computation.

There is a large body of theory about NP-completeness, but the theory only applies to worst-case analyses and to arbitrarily large problem instances. For example, the integer knapsack problem is not *strong* NP-complete, meaning that polynomial-time algorithms are available unless "exponentially large" integers are used in the problem instance [Gary79]. (Problems that are not strong in this sense are said to be solvable in *pseudo-polynomial time*.)

Two other classes of polynomial-time algorithms can be used to try to solve NP-complete problems. There are clever *approximation* algorithms which always get a approximate answer, though not necessarily the exact answer. (See [Gary79] and [Horo78] for examples.) There are also *non-deterministic* algorithms, which give an exact answer but may not give any answer at all [Horo78]. Of course there is no known polynomial-time algorithm that will solve worst-case, arbitrarily large problem instances, but algorithms like those above might force unacceptably large instances of an NP-complete problem if a PKC using it is to be secure.

There is specialized problem, *factoring*, that has become the basis for several cryptographic applications, including

- the RSA-like cryptosystem with exponent 2 [Rab79], [Will80],
- the generation of a cryptographically secure random number generator [Sha83c], and
- the exchange of secret keys without an arbiter [Blum83].

Each has a protocol with a complete security proof, assuming that factoring is intractable. And of course the RSA cryptosystem itself, while not equivalent to factoring, depends on the difficulty of factoring for its security [Riv79]. New applications that depend on factoring appear regularly [Ong84]. As long as factoring remains intractable, we are in a good position, but we are overdependent on the computational complexity of one particular problem.

The exact complexity status of the factoring problem is not known, though as with knapsack, factoring is easily solved in pseudo-polynomial time. As before, even if no polynomial-time algorithm is found, unacceptably large integers might eventually be required to keep the problem intractable.

\*       \*       \*       \*

Thus it seems natural and desirable to look toward harder problems as the basis for a PKC. There are various *provably intractable* problems [Aho74], and of course the undecidable problems for which no general algorithmic solution can exist. It is important to note that for a PKC one could use only a *special instance* of one of these harder problems. The difficulty of cryptanalysis would still be in the class NP.

This paper is the result of an initial look at various undecidable problems, trying to construct PKC's. We are concentrating on a particular problem along the lines of an earlier paper [Wag84], with more specific details included here.

## 2. THE WORD PROBLEM.

There are undecidable problems for finitely presented groups and for semigroups. First we need a number of definitions. (See [Magn66], [Rotm73], [Crow63], [Lynd77].) A

*finitely presented group*  $G$  consists of *generators*  $x_1, x_2, \dots, x_n$ , which are just abstract symbols, and *relators*  $r_1 = e, r_2 = e, \dots, r_m = e$ , to be defined below. Corresponding to each generator  $x_i$  there is an *inverse*  $x_i^{-1}$ . A *word* in  $G$  is a finite string made up of symbols  $x_i$  and  $x_i^{-1}$ . The *empty string*  $e$  is also a word, the *identity* of the group. Each of the  $r_i$  above is a word. The group operation for combining words is *concatenation*. For each word  $w$ , the *inverse word*  $w^{-1}$  consists of all the symbols of  $w$  written in reverse order, where each  $x_j$  is replaced by  $x_j^{-1}$  and each  $x_j^{-1}$  is replaced by  $x_j$ .

The group  $G$  consists of equivalence classes of all possible words. Two words  $w$  and  $v$  are *equivalent* in  $G$  if we can transform  $w$  to  $v$  by a finite sequence of *replacement rules* of the form

- Rule (i): changing  $x_i x_i^{-1}$  or  $x_i^{-1} x_i$  to  $e$ , that is eliminating  $x_i x_i^{-1}$  or  $x_i^{-1} x_i$ ,
- Rule (ii): introducing  $x_i x_i^{-1}$  or  $x_i^{-1} x_i$  at any point,
- Rule (iii): changing  $r_j$  or  $r_j^{-1}$  to  $e$ , that is eliminating  $r_j$  or  $r_j^{-1}$ ,
- Rule (iv): introducing  $r_j$  or  $r_j^{-1}$  at any point.

There is a more formal way to define these concepts. First the *free group*  $F$  on generators  $x_1, x_2, \dots, x_n$  is defined as the set of all words in the  $x_i$  and  $x_i^{-1}$  that are *reduced* by repeatedly cancelling out  $x_i x_i^{-1}$  and  $x_i^{-1} x_i$  until no further cancellations are possible. Let  $R$  be the *normal* subgroup generated by the words  $r_1, r_2, \dots, r_m$  ( $R$  is the intersection of all normal subgroups containing the  $r_i$ .) Finally  $G$  is the quotient group  $F/R$ .

The *word problem* for a group  $G$  is the decision problem that asks for each word  $w$ , whether  $w$  is equivalent to the identity of  $G$  (Equivalently one can ask whether two given words are equivalent.) It turns out that there exist *specific* groups for which the word problem is undecidable [Nov55], [Boon59], [Rab58]. Like any



undecidable problem, the word problem can only be undecidable as a question asked about infinitely many words -- any finite collection of words must have a decidable word problem.

Finitely presented groups are extremely complex objects. For example, the free group on two generators with no relators contains within it as a subgroup the free group on a countably infinite number of generators. There is a great deal of structure and theory associated with such groups and with the word problem. A number of researchers devote their entire energies to this subject [Lynd77].

There is a similar and simpler *word problem* for *semigroups*. We start with generators and words in the generators as before but without the inverses. Instead of relators we have a list of *equations* of the form  $a_1 = b_1, a_2 = b_2, \dots, a_m = b_m$ . In defining equivalent words we can only replace any occurrence of  $a_j$  by  $b_j$  and vice versa. The word problem for semigroups again asks if we can decide whether two given words are equivalent. Using the halting problem, for example, it is easy to see that there is a specific semigroup for which the word problem is undecidable.

All of our discussion of groups in this paper can be regarded as just a special case of semigroups since one could regard the group as a semigroup with extra symbols  $x_i^{-1}$  and extra equations  $x_i, x_i^{-1} = e$ , etc. Thus a semigroup would just be more general and flexible for our applications. We have chosen to emphasize groups because they seem to fit in naturally with our main ideas for cryptosystems and because of the enormous amount of research on groups and their word problems. One hopes that such a thoroughly studied problem might someday yield a good theoretical foundation for a cryptosystem.

### 3. PUBLIC-KEY CRYPTOSYSTEMS.

The word problem is similar to the knapsack problem in that both are "natural" problems for public-key cryptosystems, i.e., both immediately and directly allow public encryption. The difficulty is to insert a trapdoor that will allow decryption. (See [Diff76].)

The trapdoor then becomes a point of weakness for cryptanalytic attacks. We feel that a harder problem may make direct attacks more difficult and allow more leeway for such trapdoor insertions.

To use the word problem to encrypt a *single bit*, start with a finitely presented group  $G$  and with two "special" words  $w_1$  and  $w_2$  known to be inequivalent in  $G$ . Choose one of  $w_1$  and  $w_2$  and randomly apply Rule (i) through Rule (iv) to the word, resulting in a word  $v$  equivalent to either  $w_1$  or  $w_2$  (but not both). Thus the public key consists of the group  $G$  and the special words  $w_1$  and  $w_2$ .

This is a randomized encryption procedure in the sense of [Rive83]. There are infinitely many possible ciphertexts corresponding to each plaintext *bit*, and the system has an arbitrarily large expansion factor.

This property of a large expansion factor is not new. In fact the homophonic ciphers introduced centuries ago [Denn82] associate one of a (finite) set of ciphertext elements with each plaintext element. In this case security is increased with greater expansion factors to perfect security "when each letter of plaintext enciphers into a unique ciphertext symbol" [Denn82]. As another example Brassard [Bras81] mentions a message length  $n$  to ciphertext length  $n^2$  expansion. More generally various randomization techniques [Rive82] can be used to trade a larger expansion factor for the likelihood of increased security.

One can improve the large expansion factor in our system as follows: to encrypt  $n$  bits, select  $q = 2^n$  mutually inequivalent special words  $w_1, w_2, \dots, w_q$ . Choose one of these and encrypt as above. This gives an  $n$ -fold improvement in expansion factor, but makes the "special word" part of the public key  $2^n$  times as long.

With good choices for the group and special words in the encryption method described above, it appears that decryption can be made very difficult. Decryption difficulty also depends on the number and type of replacements made during encryption.

#### 4. TRAPDOOR INSERTION.

There are surely many ways to insert a trapdoor. We present one general approach in section 4.1. Section 4.2 discusses cryptanalysis, and section 4.3 gives a more specific approach whose implementation is discussed in section 5.

##### 4.1. GENERALITIES.

Start with a finitely presented group

$$G = \langle x_1, x_2, \dots, x_n \mid r_1 = e, r_2 = e, \dots, r_m = e \rangle,$$

and add more relators

$$s_1 = e, s_2 = e, \dots, s_p = e,$$

to get another finitely presented group  $G'$ . There is a special relationship between  $G$  and  $G'$ . In formal group theory terms if  $N$  is the normal subgroup of  $G$  generated by the words  $s_1, s_2, \dots, s_p$ , then  $G'$  is the *quotient group*  $G' = G/N$ . There is a natural function (the *quotient mapping*)  $\Omega: G \rightarrow G'$  defined as follows. If  $x$  is a group element of  $G$  (= equivalence class of words) let  $w$  be any word representing  $x$ . Then  $\Omega(x)$  is just the equivalence class of  $w$  within  $G'$ . For us the most important property of  $\Omega$  is the following: if  $x$  and  $y$  are equivalent in  $G$ , then  $\Omega(x)$  and  $\Omega(y)$  are equivalent in  $G'$ . Stated another way, if  $\Omega(x)$  and  $\Omega(y)$  are *not* equivalent in  $G'$ , then  $x$  and  $y$  must be not equivalent in  $G$ . (The converse does not hold: in fact it is easy for elements *not* equivalent in  $G$  to "collapse" to equivalent elements in  $G'$ .)

For this trapdoor to work, the  $w_1$  and  $w_2$  from  $G$  that are part of the public key must have the property that  $\Omega(w_1)$  and  $\Omega(w_2)$  are not equivalent in  $G'$ . To decrypt one needs to decide in  $G'$  which of  $\Omega(w_1)$  and  $\Omega(w_2)$  the word  $\Omega(y)$  is equivalent to, i.e., one needs to solve the word problem in  $G'$ , at least for some words. ( $\Omega(y)$  must be equivalent in  $G'$  to one or the other, but not both.)

The idea behind this method is that the word problem in  $G$  might be intractable, while the extra relators  $\{s_i = e\}$  might simplify things so that there is an efficiently solvable word problem for  $G'$ . This general decryption method would work both for the owner of the PKC and for an opponent attempting cryptanalysis. This method is a standard way in group theory to show that two elements are not equivalent.

#### 4.2. CRYPTANALYSIS.

We now list possible cryptanalytic attacks on this type of cryptosystem. Assume that a finitely presented group  $G$  is given along with two special inequivalent words  $w_1$  and  $w_2$  for the public key. We regard the quotient group  $G'$  or the extra  $s_i = e$  relators as the secret key. One of  $w_1$  or  $w_2$  is chosen and encrypted to form a word  $y$ .

*Attack (a): Find a tractable algorithm which decides the word problem in  $G$ .* With this algorithm, one directly decides which of  $w_1$  or  $w_2$  is equivalent to  $y$ . Gilles Brassard has pointed out that there is always a simple but impractical constructive algorithm that works for two words (or finitely many words): just try all possible sequences of replacements in parallel on  $w_1$  and  $w_2$ , producing the word  $y$  in a finite amount of time. Thus as we have mentioned before, in no sense is cryptanalysis undecidable. However, one hopes that with a good choice of  $G$ , there will be no tractable algorithms for direct attacks of this sort.

*Attack (b): Find extra relators  $\{s_i = e\}$  with quotient group  $G'$ , so that  $\Omega(w_1)$  is not equivalent to  $\Omega(w_2)$  and so that there is a tractable algorithm in  $G'$  to decide the word problem.* This is just the general method mentioned in section 4.1. These particular extra relators do not need to be the same as in the secret key -- just so the other conditions are satisfied. Guarding against this bothersome attack is the main reason for

the complexity of section 5.

Attack (c): *Use a brute-force attack on a PKC, in which one decrypts the ciphertext under each possible secret key.* This attack would succeed and shows that cryptanalysis is in the class NP. However, unlike the situation with a traditional PKC, here there is no fixed bound on the size of the smallest secret key that would work for decryption. There might be infinitely many possible candidate secret keys to try, and success might take arbitrarily long. (The specific system described in section 5 has only finitely many possible secret keys for each public key.)

Attack (d): *Regard the system as a conventional cryptosystem, and use a brute-force known plaintext attack in which the plaintext is encrypted under each possible key.* This attack is almost hopeless, since even with the correct key the ciphertext is not determined, and since there is no bound on the number of keys to try.

#### 4.3. DETAILS.

In order to construct a specific trapdoor, we propose choosing the additional relators  $\{s_i = e\}$  so that each of the  $r_i = e$  becomes trivial. The words  $r_i$  are also chosen to facilitate this. Consider extra relators of one of three forms:

- Type (S1): (*Elimination* of a generator)  
 $x_i = e$ , for some specific  $i$ . (Thus any occurrence of  $x_i$  just drops out.)
- Type (S2): (*Collapse* of two generators to one)  
 $x_i x_j^{-1} = e$ , or  $x_i x_j = e$ , for specific  $i$  and  $j$ . (Any reference to  $x_j$  can be replaced by  $x_i$  or by  $x_i^{-1}$ .)
- Type (S3): (*Commutator* of two generators)  
 $x_i x_j x_i^{-1} x_j^{-1} = e$ , or  $x_i x_j = x_j x_i$ , for specific  $i$  and  $j$ .  
 ( $x_i$  and  $x_j$  commute.)

Such extra relators might greatly simplify the  $r_i$  words. In fact we will choose the  $r_i$  and the  $s_j$  in such a way that each  $r_i$ , in conjunction with all the  $s_j$ , will reduce

to the empty word  $e$ . After eliminating and collapsing generators, the group  $G'$  will have *only* relators of type (S3) (the commutators). Thus  $G'$  will be a free group in which certain pairs of generators commute. There is a simple (polynomial-time) algorithm which decides the word problem for such a group. (See section 5.) Notice that in  $G'$  the special words  $w_1$  and  $w_2$  must still not be equivalent.

This method for inserting a trapdoor could also be as an attack by an opponent, a special case of Attack (b) of the previous section.

Attack (b'): Find extra relators  $\{s_i = e\}$  of types (S1), (S2) and (S3) such that

- (i) each  $r_i$  word becomes trivial, and
- (ii) even with the extra  $\{s_i = e\}$  relators, the words  $w_1$  and  $w_2$  are still not equivalent.

We propose to choose the special words  $w_1$  and  $w_2$  so that for "most" choices of  $s_i = e$  relators, condition (ii) will not be true. Here is our approach in outline form. Given a large collection of  $r_i = e$  relators the opponent or PKC originator must introduce many commuting pairs of generators in order to make all the  $r_i$  trivial. So in the simplified group  $G'$ , most pairs of generators will commute. The PKC originator will have a small (secret) subset of non-commuting pairs.

It is fairly easy to construct arbitrarily many inequivalent words that reduce to  $e$  if any one of a set of pairs commutes. For a single pair  $(x_1, x_2)$ , just use the word  $x_1 x_2 x_1^{-1} x_2^{-1}$  or  $x_1 (x_2)^2 x_1^{-1} (x_2^{-1})^2$ , etc. For a recursive general definition, assume  $u$  is a word that reduces to  $e$  in case any one of a set  $U$  of pairs commutes. Assume  $v$  and  $V$  have the same property. Then the word  $u v u^{-1} v^{-1}$  will reduce to  $e$  in case any one pair commutes from the union of the sets  $U$  and  $V$ .

In this form,  $w_1$  and  $w_2$  would still not work in a public key. Before publishing them, they must be encrypted as we have described, so that the set of commuting pairs will no longer be recognizable.

## 5. AN EXPERIMENTAL IMPLEMENTATION.

We have chosen specific parameters, relators, and special words and have written a computer program to implement an example cryptosystem. We should emphasize that we are only attempting a rough implementation to demonstrate the feasibility of this system, and to stimulate further research. Much more work will be required before anyone could rely on the security and practicality of any cryptosystem similar to this one.

The relators  $r_i = e$  are chosen so that for each generator  $x_j$  appearing in  $r_i$  there is a corresponding  $x_j^{-1}$ . Then it is easy to make each  $r_i$  trivial by allowing certain pairs to commute, i.e., adding relators of type (S3) (section 4.3). Adding relators of types (S1) and (S2) will give alternative ways to make the  $r_i$  trivial. The basic idea is to present an opponent with a very large number of ways to get rid of the  $r_i$  relators. We might hope that the opponent would have to search for the secret subset of non-commuting pairs in order to break this system.

After some searching around, we have settled on relators of three types for the original  $\{r_i = e\}$ , where below  $x_p, x_j, x_k$ , and  $x_l$  stand for arbitrary generators or inverses of generators.

Type (R1):  $x_i x_j x_k x_l x_i^{-1} x_k^{-1} x_j^{-1} x_l^{-1} = e$ ,

Type (R2):  $x_i x_j x_k x_i^{-1} x_j^{-1} x_k^{-1} = e$ , and

Type (R3):  $x_i x_j x_k x_i^{-1} x_k^{-1} x_j^{-1} = e$ .

We mostly use relators of type (R1) with some of types (R2) and (R3). We do not know the complexity of the word problem for a group made up of relators of these forms, so that Attack (a) of section 4.2 might succeed. (It would be better to start with a group  $G$  with an undecidable word problem.)

Type (R1) has the advantage that there are seven distinct ways to make such a relator vanish using a minimal number of extra relators of types (S2) (*collapsing*) and (S3) (*commutators*). (Type (S1) (*elimination*) relators are rather too drastic to use much, if at all.) For example, suppose we use an extra collapsing relator

$$x_j x_j = e, \text{ or } x_j = x_j^{-1},$$

and three extra commutators

$$x_i x_k x_i^{-1} x_k^{-1} = e, \text{ or } x_i x_k = x_k x_i$$

$$x_j x_k x_j^{-1} x_k^{-1} = e, \text{ or } x_j x_k = x_k x_j, \text{ and}$$

$$x_k x_i x_k^{-1} x_i^{-1} = e, \text{ or } x_k x_i = x_i x_k.$$

The original relator simplifies as follows

$$x_i x_j x_k x_i x_i^{-1} x_k^{-1} x_j^{-1} x_i^{-1} =$$

$$x_i x_j x_k x_j^{-1} x_i^{-1} x_k^{-1} x_j^{-1} x_j =$$

$$x_i x_j x_k x_j^{-1} x_i^{-1} x_k^{-1} =$$

$$x_i x_j x_j^{-1} x_k x_i^{-1} x_k^{-1} = x_i x_k x_i^{-1} x_k^{-1} = e.$$

There are four other very similar distinct methods to make this relator vanish. In addition, just setting  $x_j = x_j$  makes everything drop out and making five of the six possible pairs commute also makes the relator vanish.

Along similar lines there are three ways to make a relator of type (R2) vanish and two ways for a relator of type (R3). Of course any of these relators will vanish if one just allows all relevant pairs to commute, with no need to include the pair  $(x_j, x_k)$  in types (R1) or (R3).

In making up a specific PKC we have chosen four non-commuting pairs and made up special words (at least 64 symbols long) that would vanish if any one of the four pairs commuted. These pairs were chosen so that for each pair there is a specific  $r_j = e$  relator so that all but one way of making the word  $r_j$  trivial will also make the given pair commute. Thus if an opponent uses Attack (b') of section 4.3, then in making each  $r_j$  vanish, he will very likely make one or more of the crucial pairs commute, and so the special words will also vanish. (In order to keep the special word from degenerating, it was necessary to add extra non-commuting pairs.)

Applications of the replacement rules (i) through (iv) are more complicated than one might expect. For example, suppose we have a relator



$$r_1 = x_1 x_2 x_3 x_4 x_5 x_6 = e$$

and a word

$$w = x_3 x_6 x_1 x_2$$

Then in the equation  $r_1 = e$ , multiply on the left by  $x_2^{-1} x_1^{-1}$  and on the right by  $x_6^{-1}$  to get

$$x_3 x_4 x_5 = x_2^{-1} x_1^{-1} x_6^{-1}.$$

Taking the inverse of both sides gives

$$x_5^{-1} x_4^{-1} x_3^{-1} = x_6 x_1 x_2.$$

Thus in a group with the relator  $r_1 = e$ , the word  $w = x_3 x_6 x_1 x_2$  is equivalent to the word

$$x_3 x_5^{-1} x_4^{-1} x_3^{-1}.$$

(The process given above can be redone using just Rules (i) through (iv) in a formal fashion.)

In general we can write the generators of a relator clockwise in a circle, and any clockwise connected string can be replaced by the inverse of the complementary string, plus inverses of these replacements. The relator of length 6 above allows 72 different possible replacements, and a relator of length  $n$  allows  $2n^2$ . Our computer program attempts to look for replacements where the string being replaced is as long as possible. This kind of string matching can be done fairly efficiently using a variation of the Knuth-Morris-Pratt algorithm [Aho74].

In actual runs of our experimental implementation, we tried  $n = 25$  and  $n = 50$  generators. (We think the latter size might provide moderate security against attacks we can visualize.) The special words  $w_1$  and  $w_2$  are first made up as described in section 4.3, and then must be "pre-encrypted" before public release to hide the non-commuting pairs used in making them up. Actual public encryption just consists of more of the same kinds of replacements. Table 1 shows sets of parameters for two cryptosystems.

Table 1. Parameters for two experimental cryptosystems.

Number of Generators		25	50
Number of relators in $\mathcal{G}$	Type R1	34	153
	Type R2	6	21
	Type R3	6	20
	Total	46	194
Pairs of gener- tors in $\mathcal{G}$	Total	300	1225
	Non-commut.	19	29
Number of additional relators in $\mathcal{G}$	Type S1	0	0
	Type S2	3	3
	Type S3	220	1067
	Total	223	1070
Length of special word	Original	64	64
	Encrypted	$\sim 1/4$ symbol per replacem.	$\sim 1/2$ symbol per replacem.
Public key size (bits)		$\sim 1000$ - 2000	$\sim 10000$
Expansion factor (minimum)		$\sim 50$ -500	$\sim 100$ -1000

The replacements used for public encryption pose interesting problems. There need to be many random choices in the invocation of these replacements, but we do not want things completely random because we want the lengths to stay within reasonable bounds. It is also necessary that all parts of the original word get acted upon. Finally, we do not want a replacement to just undo the action of a previous replacement. To help with these goals, we maintained a "ghost" string in parallel with the real string being encrypted. The ghost keeps track of which string symbols have been replaced, using which relator. The replacement strategy was to choose a string location and relator at random, and to definitely use that relator for a replacement, trying first near the chosen location. But the algorithm was given some leeway to try to achieve the above goals. We performed thousands of replacements on the special words to get an idea of the

asymptotic behavior. With 50 generators, the last of the original symbols of the special word was replaced after about 1000 replacements. The encrypted special words were growing at the rough rate of half a symbol per replacement.

(After 2000 replacements, the special words were about 1100–1200 symbols long, though the rate of increase seemed to be slowing down.) With a better choice of the set of relators  $\{r_i = e\}$  or with more of them, encryption might not have a lengthening effect at all. We hope there is a clever way to do the design so that cryptanalysis is provably equivalent to some standard problem from combinatorial group theory, just as other systems have been proven equivalent to factoring (see section 1).

For decryption in  $G'$ , we need to solve the word problem for a free group with some commutators. The algorithm converts any word to a standard form in two phases. First consider any substring of the form  $\dots xyx^{-1}\dots$ , where  $x$  is a generator,  $y$  is a string, and  $x$  commutes with every generator in  $y$ . In such a case we cancel  $x$  and  $x^{-1}$ . Such cancellations are repeated until no more are possible. (One needs to argue that any choices along the way do not affect the final outcome.) The second part of the algorithm uses a bubblesort-type method to make sure any adjacent commuting pairs are in a standard order.

In constructing these experimental cryptosystems, most relators were just chosen at random *after* deciding on the trapdoor. With more care, one could create the trapdoor after the relators. In this way the public relators could be represented pseudo-randomly, greatly reducing the public key size.

As part of the experiment, we simulated one simple attack by the opponent. For each type R1 relator we made five of the six pairs commute so that the relator would vanish, and similarly for types R2 and R3. Then of course several of the crucial pairs used in the special words commuted, so that these special words just reduced to  $e$ . We hope that it would be an intractable problem for the opponent to achieve any other result. There are various brute-force searches that the opponent could try, but each such search, for 50 generators, seems to involve more than  $10^9$  possibilities.

## 6. CONCLUSIONS.

We have made a case for basing cryptosystems on problems harder than NP-complete. As an illustration, we have used the undecidable word problem for groups to design a public-key cryptosystem. Public encryption is straightforward, but trapdoor insertion requires further study. An experimental system was implemented and seems resistant to initial cryptanalytic attacks. This system has a large key size and encryption time, and an excessively large expansion factor, at least 100 to 1.

## ACKNOWLEDGMENT.

Dorothy Denning pointed out Brassard's work, and Jim Anderson suggested looking at homophonic ciphers. Mark Cain gave valuable help with the programming. Whit Diffie and Gilles Brassard made comments during the presentation that are incorporated into this paper.

## REFERENCES.

- [Adle83] L. M. Adleman, "On breaking the iterated Merkle-Hellman public-key cryptosystem," *Advances in Cryptology: Proceedings of Crypto 82*, ed. by D. Chaum et al., Plenum, 1983, pp. 303-308.
- [Aho74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.
- [Blum83] M. Blum, "How to exchange (secret) keys," *ACM Transactions on Computer Systems* 1, 2 (May 1983), pp. 175-193.
- [Boon59] W. W. Boone, "The word problem," *Annals of Math.* 70 (1981), pp. 207-265.
- [Bras79] G. Brassard, "A note on the complexity of cryptography," *IEEE Transactions on Information Theory*, IT-25, 2 (Mar. 1979), pp. 232-233.
- [Bras81] G. Brassard, "An optimally secure relativized cryptosystem," *Advances in Cryptography: A report on CRYPTO 81*, ed. by A. Gersho, ECE REPT. No. 82-04, Dept. of Elect. and Computer Eng., Univ. of Calif., Santa Barbara, pp. 54-58.
- [Crow63] R. H. Crowell, and R. H. Fox, *Introduction to Knot Theory*, Blaisdell, 1963.

- [Diff76] W. Diffie, and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory* IT-22, 6 (Nov. 1976), pp. 644-654.
- [Gary79] M. R. Gary, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
- [Horo78] E. Horowitz, and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.
- [Lynd77] R. C. Lyndon, and P. E. Schupp, *Combinatorial Group Theory*, Springer, 1977.
- [Magn66] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*, J. Wiley (Interscience), 1966.
- [Merk78] R. C. Merkle, and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory* IT-24, 5 (Sept. 1978), pp. 525-530.
- [Novi55] P. S. Novikov, "On the algorithmic unsolvability of the word problem in group theory," *Trudy Mat. Inst. Steklov* 44, 143 (1955).
- [Ong84] H. Ong, C. P. Schnorr, and A. Shamir, "An efficient signature scheme based on quadratic equations," *Proc. of the Sixteenth Annual ACM Symposium of Theory of Computing*, ACM 1984, pp. 208-216.
- [Rab158] M. O. Rabin, "Recursive unsolvability of group theoretic problems," *Annals of Math.* 67 (1958), pp. 172-194.
- [Rab179] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Technical Report No. TR-212, MIT Lab. for Computer Science (Jan. 1979).
- [Rive78] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* 21, 2 (Feb. 1978), pp. 120-126.
- [Rive79] R. L. Rivest, "Critical remarks on 'Critical remarks on some public-key cryptosystems'," *BIT* 19 (1979), pp. 274-275.
- [Rive83] R. L. Rivest, and A. T. Sherman, "Randomized encryption techniques," *Advances in Cryptology: Proceedings of Crypto 82*, ed. by D. Chaum et al., Plenum, 1983, pp. 145-163.
- [Rotm73] J. J. Rotman, *Theory of Groups: An Introduction*, Second Edition, Allyn and Bacon, 1973.
- [Sha83a] A. Shamir, "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," *Advances in Cryptology: Proceedings of Crypto 82*, ed. by D. Chaum et al., Plenum, 1983, pp. 279-288.
- [Sha83b] A. Shamir, "The strongest knapsack-based cryptosystem?" (presentation at Crypto 82).

- [Sha83c] A. Shamir, "On the generation of cryptographically strong pseudorandom sequences," *ACM Transactions on Computer Systems* 1, 1 (Feb. 1983), pp. 38-44.
- [Tarj83] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, 1983.
- [Wagn84] N. R. Wagner, "Searching for public-key cryptosystems," *Proceedings of the 1984 Symposium on Security and Privacy*, IEEE Computer Society, pp. 91-98.
- [Will80] H. C. Williams, "A modification of the RSA public-key encryption procedure," *IEEE Transactions on Information Theory*, IT-26, 6 (Nov. 1980), pp. 726-729.

# EFFICIENT SIGNATURE SCHEMES BASED ON POLYNOMIAL EQUATIONS

(preliminary version)

H. Ong<sup>1</sup>, C.P. Schnorr<sup>1</sup>, A. Shamir<sup>2</sup>

<sup>1</sup>Fachbereich Mathematik  
Universität Frankfurt

<sup>2</sup>Applied Mathematics Department  
The Weizman Institute of Science  
Rehovot 76100, Israel

## ABSTRACT

Signatures based on polynomial equations modulo  $n$  have been introduced by Ong, Schnorr, Shamir [3]. We extend the original binary quadratic OSS-scheme to algebraic integers. So far the generalised scheme is not vulnerable by the recent algorithm of Pollard for solving  $s_1^2 + k s_2^2 = m \pmod{n}$  which has broken the original scheme.

## 1. INTRODUCTION

Diffie and Hellman [1] introduced the concept of digital signature and that of public key cryptosystem. The RSA system [6] is currently believed to be the most secure scheme for both purposes. A new type of signature scheme based on the quadratic equation  $s_1^2 + k s_2^2 = m \pmod{n}$  has been proposed by Ong, Schnorr, Shamir [3]. Here  $m$  is the message,  $s_1$  and  $s_2$  are the signature, and  $k$  and  $n$  are the publicly known key. The new scheme would be much easier to implement than the RSA-scheme, but it has been broken by a recent algorithm of Pollard which solves the equation  $x^2 + k y^2 = m \pmod{n}$  without factoring  $n$ .

In this paper we consider signature schemes based on more general polynomial equations modulo  $n$ . In particular we extend the original OSS-scheme from rational integers to algebraic integers. This leads to a signature scheme based on the quadric equation  $(m_2 - 2ks_{12}s_{21})^2 + 4s_{22}^2(ds_{12}^2 + k(s_{21}^2 + ds_{22}^2) - m_1) = 0 \pmod{n}$  where  $m_1$  and  $m_2$  are the message,  $s_{12}$ ,  $s_{21}$  and  $s_{22}$  are the signature, and the public key consists of the integers  $k, d, n$  with  $1 \leq k, d < n$ . The private key is the square root  $\sqrt{-k} \pmod{n}$ . Signature verification can be done with 10 multiplications on integers modulo  $n$ , signature generation requires 9 multiplications and 1 division modulo  $n$ .

All participants of the system may share the  $(d, n)$ -part of their public key provided that the factorisation of  $n$  is completely unknown.

## 2. SIGNATURES BASED ON POLYNOMIAL EQUATIONS

When Alice joins the communication network she publishes a key consisting of two parts: a modulus  $n$  and the integer coefficients of a polynomial  $P(s_1, \dots, s_d) \in \mathbb{Z}[s_1, \dots, s_d]$  with indeterminates  $s_1, \dots, s_d$ . The modulus  $n$  is the product of two large random primes  $p, q$ . The factorization of  $n$  should be unknown, except possibly to Alice. In order to prevent factoring of  $n$  by known factoring algorithms  $n$  should be at least 600 bits long. The coefficients of  $P$  are integers in the range  $\mathbb{Z}_n := \{c \in \mathbb{Z} : 0 \leq c < n\}$ . The elements in  $\mathbb{Z}_n$  are used as representatives for the ring  $\mathbb{Z}/n\mathbb{Z}$  of integers modulo  $n$ . Typically  $P$  will only have a few coefficients.

The messages  $m$  are numbers in  $\mathbb{Z}_n$ . A tuple  $\underline{s} = (s_1, \dots, s_d)$  of numbers in the same range is a signature for  $m$  if it satisfies the equation

$$(1) \quad P(s_1, \dots, s_d) = m \pmod{n}.$$

Given the coefficients of  $P$  and  $n$  it is easy to verify Alice's signatures by evaluating  $P(s_1, \dots, s_d)$  with a few modular multiplications and additions.

Unlike the RSA system, signatures are not uniquely associated with messages. Since the number of possible messages is  $n$  while the number of possible signature tuples is  $n^d$ , each message has about  $n^{d-1}$  different signatures. However, the probability that a randomly chosen tuple  $\underline{s} = (s_1, \dots, s_d)$  will be a valid signature of a given  $m$  is negligible, and thus the multiplicity of signatures does not imply that they are easy to find.

The secret that helps Alice solve the equation (1) is an integer  $(d, d)$ -matrix  $A$  which modulo  $n$  is invertible. If the transformation



$\underline{x} = A\underline{s} \pmod{n}$  transforms  $P$  into a polynomial  $x_1 P'(x_2, \dots, x_d) = P(s_1, \dots, s_d) \pmod{n}$  then Alice can easily solve equation (1). She picks random values  $x_2, \dots, x_d \in \mathbb{Z}_n$ , evaluates

$$(2) \quad x_1 := m/P'(x_2, \dots, x_d) \pmod{n}$$

and transforms

$$(3) \quad \underline{s} := A^{-1} \underline{x} \pmod{n}.$$

So Alice can generate signatures of  $m$  by choosing random values  $x_2, \dots, x_d$  and evaluating (2), (3) using a few modular multiplications and additions and one modular division. If  $P'(x_2, \dots, x_d)$  is not relatively prime to  $n$  then  $m/P'(x_2, \dots, x_d) \pmod{n}$  may be not defined, but if all the factors of  $n$  are large Alice is unlikely to choose such values  $x_2, \dots, x_d$ .

The relationship between messages and signatures are summarized in the following lemma. Let  $\mathbb{Z}_n^*$  be the set of numbers in  $\mathbb{Z}_n$  which are relatively prime to  $n$ . Note that  $\mathbb{Z}_n$  represents the set  $\mathbb{Z}/n\mathbb{Z}$  of integers modulo  $n$ , so  $\mathbb{Z}_n$  is a commutative ring under addition and multiplication modulo  $n$  and  $\mathbb{Z}_n^* \subset \mathbb{Z}_n$  is the group of invertible elements.

**LEMMA 1** For every  $m \in \mathbb{Z}_n^*$  the set of signatures of  $m$  is in 1-1 correspondence with the set of values (3) as  $x_2, \dots, x_d$  range over  $\mathbb{Z}_n$  and  $x_1 = m/P'(x_2, \dots, x_d) \pmod{n}$ .

**PROOF** For every  $(x_2, \dots, x_d) \in (\mathbb{Z}_n)^{d-1}$  with  $P'(x_2, \dots, x_d) \in \mathbb{Z}_n^*$  (2), (3) clearly define a signature  $\underline{s}$  of  $m$ . On the other hand for every signature  $\underline{s} = (s_1, \dots, s_d)$  there exists  $\underline{x} := A\underline{s} \pmod{n}$ . We have  $P(s_1, \dots, s_d) = x_1 P'(x_2, \dots, x_d) = m \pmod{n}$ , and  $P'(x_2, \dots, x_d) \in \mathbb{Z}_n^*$  follows from the assumption  $m \in \mathbb{Z}_n^*$ . Since  $A$  is non singular only one value of  $(x_2, \dots, x_d) \in (\mathbb{Z}_n)^{d-1}$  can correspond to each signature. Q.E.D.

**REMARKS** (i) By using independent random values  $x_2, \dots, x_d$ , Alice can choose an arbitrary signature of  $m$  with uniform probability distribution, and is not restricted to signatures of some special form.

(ii) If several messages  $m^i$  are signed with the same  $x_2, \dots, x_d$  then  $\underline{x}^i = (x_1^i, \dots, x_d^i)$  and the signature  $\underline{s}^i$  are known for each message and  $A$  can be computed from the linear equations  $\underline{x}^i = A\underline{s}^i \pmod{n}$ . Thus Alice must choose independent random values  $x_2, \dots, x_d$  for each message.

How does Alice generate her public key? She first chooses the modulus  $n$  as a large composite number which is difficult to factor. By using a probabilistic primality testing algorithm on random integers with at least 300 bits, Alice can find after a few hundred tests two numbers  $p$  and  $q$  which are almost certainly primes. The product  $n$  of  $p$  and  $q$  is easy to compute, but even the fastest known factoring algorithm on the fastest available computer will take millions of years to

factor it. The generation of  $n$  can be done within a few hours on a typical microcomputer. Such an overnight initialization is acceptable in most applications, but if the user cannot afford it, there is a faster alternative: If a trusted third party (the NBS?) computes  $n$  and then erases  $p$  and  $q$ , no one knows the factorization of  $n$  and thus everyone can use it as a standard modulus.

In order to generate the polynomial  $P$ , Alice chooses a simple polynomial  $P'(x_2, \dots, x_d)$  with integer coefficients and then picks a random integer  $(d, d)$ -matrix  $A$ . Alice keeps  $A$  secret, transforms the polynomial  $x_1 P'(x_2, \dots, x_d)$  with  $\underline{x} := A \underline{s} \pmod{n}$  into a polynomial  $P$ ,  $P(s_1, \dots, s_d) = x_1 P'(x_2, \dots, x_d) \pmod{n}$  and publishes the coefficients of the transformed polynomial  $P$ .  $P$  is no longer linear in any of the variables. The equation  $P(s_1, \dots, s_d) = m \pmod{n}$  is apparently difficult. Alice also verifies that  $A$  is invertible modulo  $n$ . If the prime factors of  $n$  are large then singular matrices are unlikely to occur. It is important that Alice can generate  $P$  without knowing the factors of  $n$ . All the participants of the communication network may use the same simple polynomial  $P'(x_2, \dots, x_d)$  and even the same modulus  $n$  (provided that the factors of  $n$  are unknown) and differ only in their choice of  $A$ .

The security of the scheme requires to choose particular transformation matrices  $A$  which cannot be easily computed from the coefficients of  $P$  and  $P'$ . We choose the polynomial  $P'$  and the matrix  $A$  so that recovering  $A$  from the polynomials  $P$  and  $P'$  is as hard as factoring  $n$ . Since Alice is not restricted to signatures of some special form it is impossible to obtain information on the secret parameters,  $A$  and the factors of  $n$  by analysing her signatures. Also Alice herself may be unaware of the factors of  $n$ . Since Bob cannot benefit from Alice's signatures and cannot use her method for solving equation (1), he must come up with an alternative way of solving this equation. So for each class of transformations  $A$  and for each polynomial  $P'$  one must carefully analyse whether equation (1) is sufficiently difficult for the corresponding polynomials  $P$ .

The security of the scheme is based on the difficulty of factoring  $n$ . When the factors  $p$  and  $q$  of  $n$  are known the equation (1) can be solved efficiently. The probabilistic root finding algorithm of Rabin<sup>5</sup> computes  $\underline{s}', \underline{s}'' \in \mathbb{Z}^d$  such that  $P(\underline{s}') = m \pmod{p}$  and  $P(\underline{s}'') = m \pmod{q}$ . By the Chinese remainder theorem  $\underline{s}'$  and  $\underline{s}''$  can be combined to a solution  $\underline{s} = \sigma \underline{s}' + \tau \underline{s}'' \pmod{n}$ . Here  $\sigma$  and  $\tau$  are integers

$$\text{satisfying } \sigma = \begin{cases} 1 & \pmod{p} \\ 0 & \pmod{q} \end{cases}, \quad \tau = \begin{cases} 0 & \pmod{p} \\ 1 & \pmod{q} \end{cases}.$$

The binary quadratic scheme: The simplest polynomial equation (1) appears for  $d = 2$ , we transform the equation

$$(5) \quad x_1 \cdot x_2 = m \pmod{n}$$

using an arbitrary  $u \in \mathbb{Z}_n^*$  by the linear substitution

$$x_1 := s_1 + u^{-1}s_2 \pmod{n}$$

$$x_2 := s_1 - u^{-1}s_2 \pmod{n}.$$

This yields  $x_1 \cdot x_2 = s_1^2 - u^{-2}s_2^2 = m \pmod{n}$ . So the trivial equation (5) is transformed into the less trivial polynomial equation

$$(6) \quad s_1^2 + k s_2^2 = m \pmod{n} \quad \text{with } k = -u^{-2} \pmod{n}.$$

The public key of the corresponding signature scheme consists of  $n$  and  $k$ , and the private key is  $u$ . A pair  $(s_1, s_2) \in (\mathbb{Z}_n)^2$  is a valid signature for  $m$  if  $s_1^2 + k s_2^2 = m \pmod{n}$ . Recovering the private key  $u$  from the public key  $k$  requires the computation of  $\sqrt{-k} \pmod{n}$  and thus is as hard as factoring  $n$ .

Unfortunately this case of our signature concept is insecure due to a recently discovered algorithm of Pollard<sup>4</sup> which efficiently solves quadratic equations  $s_1^2 + k s_2^2 = m \pmod{n}$ . Pollard's method does not solve general polynomial equations modulo  $n$  nor does it extend to systems of polynomial equations.

### 3. THE BINARY QUADRATIC SCHEME OVER ALGEBRAIC INTEGERS

The binary quadratic scheme may still yield a good signature scheme if we replace rational integers  $x_1, x_2, s_1, s_2, m$  by algebraic integers  $X_1, X_2, S_1, S_2, M$  which range over the set

$$\mathbb{Z}_{n,d} := \{a + b\sqrt{d} \mid a, b \in \mathbb{Z}, 0 \leq a, b < n\}.$$

The set  $\mathbb{Z}_{n,d}$  can play a similar role as the set  $\mathbb{Z}_n$  of integers modulo  $n$ . There is a natural way of adding and multiplying elements in  $\mathbb{Z}_{n,d}$ :

$$(a' + b'\sqrt{d}) + (a'' + b''\sqrt{d}) := a + b\sqrt{d}$$

$$\text{with } a := a' + a'' \pmod{n}, \quad b := b' + b'' \pmod{n}$$

$$(a' + b'\sqrt{d})(a'' + b''\sqrt{d}) = a + b\sqrt{d}$$

$$\text{with } a := a'a'' + db'b'' \pmod{n}, \quad b := a'b'' + a''b' \pmod{n}.$$

So all arithmetic operations in  $\mathbb{Z}_{n,d}$  are done modulo  $n \mathbb{Z}[\sqrt{d}]$  and in standard algebraic notation  $\mathbb{Z}_{n,d}$  is the ring  $\mathbb{Z}[\sqrt{d}]/n \mathbb{Z}[\sqrt{d}]$ . An element  $a + b\sqrt{d}$  is invertible iff  $a^2 - b^2d \in \mathbb{Z}_n^*$ , and in this case

$$(a + b\sqrt{d})^{-1} = a' - b'\sqrt{d}$$

with  $a' = a(a^2 - b^2d)^{-1} \pmod{n}$ ,  $b' = b(a^2 - b^2d)^{-1} \pmod{n}$ . Let  $\mathbb{Z}_{n,d}^* \subset \mathbb{Z}_{n,d}$  be the subgroup of invertible elements.

**LEMMA 2** With the above arithmetic operations  $\mathbb{Z}_{n,d}$  forms a commutative ring with  $\mathbb{Z}_n \subset \mathbb{Z}_{n,d}$ ,  $\mathbb{Z}_n^* \subset \mathbb{Z}_{n,d}^*$ .

In the sequel we let the variables  $X_1, X_2, S_1, S_2, M$  range over  $\mathbb{Z}_{n,d}$ . For an arbitrary  $u \in \mathbb{Z}_n^*$  the substitution

$$(7) \quad \begin{aligned} X_1 &:= S_1 - u^{-1}S_2 \\ X_2 &:= S_1 + u^{-1}S_2 \end{aligned}$$

yields  $X_1X_2 = S_1^2 + kS_2^2$  with  $k := -u^{-2} \pmod{n}$ . So given  $u$  the equation

$$(8) \quad X_1X_2 = M$$

which can easily be solved for any  $M \in \mathbb{Z}_{n,d}$ , is equivalent to the less trivial equation

$$(9) \quad S_1^2 + kS_2^2 = M.$$

This observation yields an efficient signature scheme. For key generation Alice picks a random element  $u \in \mathbb{Z}_n^*$  publishes  $k := -u^{-2} \pmod{n}$ , and keeps  $u$  secret. For any  $M$  Alice can easily solve the equation (9). She picks  $X_1 \in \mathbb{Z}_{n,d}^*$  at random, computes  $X_2 := MX_1^{-1}$  and inverts the linear substitution (7)

$$\begin{aligned} S_1 &:= (X_2 + X_1)/2 \\ S_2 &:= (X_2 - X_1)u/2. \end{aligned}$$

Once  $k$  is published, Bob (or anyone else) cannot compute  $u$ , and cannot follow the method of solving equation (9) that Alice is using.

For convenience we write polynomial equations over  $\mathbb{Z}_{n,d}$  as systems of polynomial equations over  $\mathbb{Z}_n$ . Let

$$\begin{aligned} X_i &= x_{i1} + \sqrt{d} x_{i2} & i = 1, 2 \\ S_i &= s_{i1} + \sqrt{d} s_{i2} & i = 1, 2 \\ M &= m_1 + \sqrt{d} m_2 \end{aligned}$$

with  $x_{ij}, s_{ij}, m_i \in \mathbb{Z}_n$ . The equation  $X_1 \cdot X_2 = M$  can be written as

$$(10) \quad \begin{aligned} x_{11} x_{21} + d x_{12} x_{22} &= m_1 \pmod{n} \\ x_{11} x_{22} + x_{12} x_{21} &= m_2 \pmod{n}. \end{aligned}$$

The equation  $S_1^2 + kS_2^2 = M$  can be written as

$$(11) \quad \begin{aligned} s_{11}^2 + d s_{12}^2 + k(s_{21}^2 + d s_{22}^2) &= m_1 \pmod{n} \\ 2(s_{11}s_{22} + k s_{12}s_{21}) &= m_2 \pmod{n}. \end{aligned}$$

Elimination of  $s_{11}$  in the latter equation yields

$$s_{11} = (m_2 - 2k s_{12} s_{21}) / (2 s_{22}) \pmod{n}$$

Therefore the system of equations (11) is equivalent to the ternary, quadric equation (12) provided that  $s_{22} \in \mathbb{Z}_n^*$ :

$$(12) \quad (m_2 - 2k s_{12} s_{21})^2 + 4 s_{22}^2 (d s_{12}^2 + k(s_{21}^2 + d s_{22}^2) - m_1) = 0 \pmod{n}.$$

So this equation can be taken as verification condition for the binary quadratic signature scheme over  $\mathbb{Z}_{n,d}$ .

The signature scheme based on equation (12) consists of the following components:

#### Key generation

1. choose two random primes  $p, q$  so that  $p \cdot q$  is difficult to factor, put  $n := p \cdot q$ .
2. pick random integers  $u, d$  which are relatively prime to  $n$ .
3. publish  $k := -u^{-2} \pmod{n}$ ,  $d$ ,  $n$ , and keep  $u$  secret.

Messages are pairs  $(m_1, m_2)$  of integers in the range  $0 < m_1, m_2 < n$ , i.e.  $m_1, m_2 \in \mathbb{Z}_n - 0$ .

#### Signature verification

A triple  $(s_{12}, s_{21}, s_{22})$  of integers in  $\mathbb{Z}_n$  is a valid signature for the message  $(m_1, m_2)$  if it satisfies the equation (12)

$$(m_2 - 2k s_{12} s_{21})^2 + 4 s_{22}^2 (d s_{12}^2 + k(s_{21}^2 + d s_{22}^2) - m_1) = 0 \pmod{n}.$$

This equation can easily be checked using  $k, d, n$  with 10 multiplications, 4 additions/subtractions modulo  $n$ . We do not count the trivial multiplication by 4.

#### Signature generation

(We solve the easy system (10), and using the private key  $u$  we transform its solution into a solution of (12) by inverting the linear substitution (7).)

1. pick random elements  $x_{11}, x_{12} \in \mathbb{Z}_n$  so that  $x_{11}^2 - d x_{12}^2$  is relatively prime to  $n$ .

$$2. \quad x_{22} := \frac{m_2 x_{11} - m_1 x_{12}}{x_{11}^2 - d x_{12}^2} \pmod{n},$$

$$3. \quad x_{21} := \frac{(m_1 x_{11} - d m_2 x_{22})}{x_{11}^2 - d x_{12}^2} \pmod{n}$$

$$4. \quad s_{12} := (x_{22} + x_{12})/2 \pmod{n}$$

$$s_{21} := (x_{21} - x_{11}) u/2 \pmod{n}$$

$$s_{22} := (x_{22} - x_{12}) u/2 \pmod{n}$$

LEMMA 3 Signature generation can be done with 9 multiplications, 1 division modulo  $n$ . (The division by 2 is trivial).

PROOF Compute  $x_{11}^2, dx_{12}, dx_{12}^2, dm_2x_{12}$  with only 4 multiplications modulo  $n$ . Obviously the rest of the computation can be done with 5 multiplications and 1 division modulo  $n$ . Q.E.D.

For a message  $(m_1, m_2)$  let  $M := m_1 + m_2\sqrt{d}$  be the corresponding element in  $\mathbb{Z}_{n,d}$ , obviously  $m_1^2 - dm_2^2 \in \mathbb{Z}_n^*$  iff  $M \in \mathbb{Z}_{n,d}^*$ . For messages  $(m_1, m_2)$  with  $m_1^2 - dm_2^2 \in \mathbb{Z}_n^*$  the above signature procedure generates arbitrary signatures of  $(m_1, m_2)$  with uniform probability distribution.

LEMMA 4 For every message  $(m_1, m_2)$  with  $m_1 + m_2\sqrt{d} \in \mathbb{Z}_{n,d}^*$  the set of signatures of  $(m_1, m_2)$  is in 1-1 correspondence with the set of values  $(s_{12}, s_{21}, s_{22})$  in step 4, as  $x_{11} + x_{12}\sqrt{d}$  ranges over  $\mathbb{Z}_{n,d}^*$ .

PROOF The set of signatures of  $(m_1, m_2)$  is in 1-1 correspondence to the set of solutions  $(S_1, S_2)$  of  $S_1^2 + kS_2^2 = M$ . By the linear transformation (7) the set of solutions  $(S_1, S_2)$  of  $S_1^2 + kS_2^2 = M$  is in 1-1 correspondence to the set of solutions  $(X_1, X_2)$  of  $X_1 \cdot X_2 = M$ . Since  $M \in \mathbb{Z}_{n,d}^*$  these solutions are in 1-1 correspondence with the set of elements  $X_1 \in \mathbb{Z}_{n,d}^*$  (remember that  $X_1 = x_{11} + x_{12}\sqrt{d} \in \mathbb{Z}_{n,d}^*$  iff  $x_{11}^2 - dx_{12}^2$  is relatively prime to  $n$ ). Q.E.D.

As a consequence of Lemma 4 messages  $(m_1, m_2)$  for which  $m_1^2 - dm_2^2$  is not relatively prime to  $n$  should be avoided. We have excluded messages with  $m_1 = 0$  or  $m_2 = 0$  anyway, see remark 7 (iv), (v). No other message  $(m_1, m_2)$  with  $\gcd(m_1^2 - dm_2^2, n) \neq 1, n$  is likely to occur.

REMARKS 5 The characteristical properties of the original binary quadratic OSS-scheme remain intact: i) The generation of the keys  $u, k := -u^2 \pmod{n}$ ,  $d$  can be done without knowing the factorization of  $n$ . All public keys may share the  $(d, n)$ -part provided that the factorization of  $n$  is unknown to all participants of the system. ii) Computing the private key  $u$  from the public key  $k, n$  requires to compute  $\sqrt{-k} \pmod{n}$ , and thus is as hard as factoring  $n$ . iii) The signature scheme is multiplicative over  $\mathbb{Z}_{n,d}$ . Solutions  $S_1', S_2'$  and  $S_1'', S_2''$  of

$$S_1'^2 + kS_2'^2 = M', \quad S_1''^2 + kS_2''^2 = M''$$

yield a solution  $S_1, S_2$  of  $S_1^2 + kS_2^2 = M'M''$  as

$$S_1 = S_1'S_1'' - kS_2'S_2'', \quad S_2 = S_1'S_1'' - kS_2'S_2''$$

iv) The roles of  $k, M$  in the equation  $S_1^2 + kS_2^2 = M$  can be interchanged since  $S_1^2 + kS_2^2 = M$  is equivalent to  $(S_1/S_2)^2 - MS_2^{-2} = -k$ .

With these remarks the following theorem can be proved in the same way as its counterpart in [3].

THEOREM 6 Any algorithm for computing  $u$  from random signatures of messages of its choice can be transformed into a probabilistic factor-

ring algorithm with similar complexity.

PROOF see proof of theorem 2 [3].

REMARKS 7 i) The theorem can easily be extended to the case of an algorithm that succeeds for only some of the  $u$ -values provided that the fraction of these  $u$ -values is non negligible. ii) In Rabin's signature scheme an opponent can factor  $n$  by analysing the signature of specific messages. In our scheme the factorization of  $n$  and the secret parameter  $u$  cannot be revealed by chosen message attacks. iii) If Bob could compute one of the  $x_{ij}$ -values  $i, j \in \{0, 1\}$  corresponding to a signature  $s_{ij}$   $i, j \in \{0, 1\}$ , he could compute  $u$ . For instance given  $x_{11}$ ,  $s_{11}$  and  $s_{21}$ , Bob can compute  $u$  from  $x_{11} = s_{11} - u^{-1} s_{21} \pmod{n}$ . A single  $x_{ij}$ -value is thus as hard to compute as  $u$ . iv) Messages  $(m_1, m_2)$  with  $m_2 = 0$  can be signed without the private key  $u$ . It is sufficient to solve

$$s_{11}^2 + k s_{21}^2 = m_1 \pmod{n}$$

by Pollard's algorithm [4]. v) Messages  $(m_1, m_2)$  with  $m_1 = 0$  can also be signed without the private key  $u$ . This easily follows from (iii) and the multiplicativity of the scheme (remark 5, iii).

THE COMPLEXITY OF SOLVING  $s_1^2 + k s_2^2 = M$  over  $\mathbb{Z}_{n,d}$

Pollard [4] solves the equation  $s_1^2 + k s_2^2 = m \pmod{n}$  by successively reducing  $m$  and  $k$ . He reduces  $m$  to  $m' \leq \sqrt{k}$ , interchanges  $m$  and  $k$ , and continues until both  $m$  and  $k$  are 1. His basic reduction step uses the euclidean algorithm over  $\mathbb{Z}$ .

Pollard's method does not solve  $s_1^2 + k s_2^2 = M$  since  $\mathbb{Z}[\sqrt{d}]$  is not euclidean domain provided that  $d > 73$  or  $d < -11$ . In particular there exist  $A, B \in \mathbb{Z}[\sqrt{d}]$  such that  $|N(A - C \cdot B)| > |N(B)|$  for all  $C \in \mathbb{Z}[\sqrt{d}]$ , (where  $N$  is the norm,  $N(x + \sqrt{d}y) = x^2 - dy^2$ ). It is unlikely that the missing euclidean algorithm for  $\mathbb{Z}[\sqrt{d}]$  can be replaced by some other norm reducing procedure. For large  $|d|$  almost all elements  $A \in \mathbb{Z}[\sqrt{d}]$  with  $|N(A)| \ll d$  are rational integers and these are unlikely to appear in a general procedure over  $\mathbb{Z}[\sqrt{d}]$ .

The methods for solving  $s_1^2 + k s_2^2 = m \pmod{n}$  which use the class group of quadratic form with discriminant  $-4k$ , see [3], do not solve  $s_1^2 + k s_2^2 = M$ . The reason is that equivalence classes of quadratic forms with coefficients in  $\mathbb{Z}[\sqrt{d}]$  cannot be represented in a canonical way by reduced forms.

The fastest known method for solving  $s_1^2 + k s_2^2 = M$  is by factoring  $n$ . This method becomes infeasible if  $n$  is at least 600 bits long.

The complexity of solving general polynomial equations modulo  $n$  is

an open problem and it may become an important subject for further cryptographic research.

#### REFERENCES

1. Diffie, W. and Hellman, M.: New Directions in Cryptography. IEEE, IT-22, (1976), 644-654.
2. Ong, H. and Schnorr, C.P.: Signatures Through Approximate Representations by Quadratic Forms. Advances in Cryptology: Proceedings of Crypto 83. Plenum Publ. New York 1984, 117-132.
3. Ong, H., Schnorr, C.P., and Shamir, A.: An Efficient Signature Scheme Based on Quadratic Equations. Proceedings of 16th ACM-Symp. of Theory of Computing, Washington (1984), p. 208-216.
4. Pollard, J.M.: Solution of  $x^2 + ky^2 \equiv m \pmod{n}$ , with Application to Digital Signatures. Preprint 1984.
5. Rabin, M.O.: Probabilistic Algorithms in Finite Fields. SIAM J. on Computing 9 (1980), p. 273-280.
6. Rivest, R.L., Shamir, A. and Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Comm. ACM 21 (1978) 120-126.
7. Shamir, A.: Identity Based Cryptosystems & Signature Schemes. Proceedings of Crypto 84.



## IDENTITY-BASED CRYPTOSYSTEMS AND SIGNATURE SCHEMES

Adi Shamir

Department of Applied Mathematics  
The Weizmann Institute of Science  
Rehovot, 76100 Israel

### THE IDEA

In this paper we introduce a novel type of cryptographic scheme, which enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. The scheme assumes the existence of trusted key generation centers, whose sole purpose is to give each user a personalized smart card when he first joins the network. The information embedded in this card enables the user to sign and encrypt the messages he sends and to decrypt and verify the messages he receives in a totally independent way, regardless of the identity of the other party. Previously issued cards do not have to be updated when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an indefinite period.

The scheme is ideal for closed groups of users such as the executives of a multinational company or the branches of a large bank, since the headquarters of the corporation can serve as a key generation center that everyone trusts. The scheme remains practical even on a nationwide scale with hundreds of key generation centers and millions of users, and it can be the basis for a new type of personal identification card with which everyone can electronically sign checks, credit card slips, legal documents, and electronic mail.

The scheme is based on a public key cryptosystem with an extra twist: Instead of generating a random pair of public/secret keys and publishing one of these keys, the user chooses his name and network address as his public key. Any combination of name, social security number, street address, office number or telephone number can be used (depending on the context) provided that it uniquely identifies the user in a way he cannot later deny, and that it is readily available to the

other party. The corresponding secret key is computed by a key generation center and issued to the user in the form of smart card when he first joins the network. The card contains a microprocessor, an I/O port, a RAM, a ROM with the secret key, and programs for message encryption/decryption and signature generation/verification.

An identity-based scheme resembles an ideal mail system: If you know somebody's name and address you can send him messages that only he can read, and you can verify the signatures that only he could have produced. It makes the cryptographic aspects of the communication almost transparent to the user, and it can be used effectively even by laymen who know nothing about keys or protocols.

When user A wants to send a message to user B, he signs it with the secret key in his smart card, encrypts the result by using B's name and network address, adds his own name and network address to the message, and sends it to B. When B receives the message, he decrypts it using the secret key in his smart card, and then verifies the signature by using the sender's name and network address as a verification key.

The secret keys must be computed by a key generation center rather than by the users, since there is nothing special about a user's identity: If user A could compute the secret key that corresponds to the public key "A", he could also compute the secret keys that correspond to the public keys "B", "C", etc., and the scheme would not be secure. The key generation center can be in a privileged position by knowing some secret information (such as the factorization of a large number) which enables it to compute the secret keys of all the users in the network.

The overall security of the scheme depends on the following points: (a) The security of the underlying cryptographic functions. (b) The secrecy of the privileged information stored at the key generation centers. (c) The thoroughness of the identity checks performed by the centers before they issue cards to users. (d) The precautions taken by users to prevent the loss, duplication, or unauthorized use of their cards.

The cryptographic scheme effectively ties the message with the identification information  $i$ , and the ownership of the card effectively ties  $i$  with the physical user. Like any other agency that issues ID cards, the center must carefully screen requests for cards to prevent misrepresentations, and must carefully protect its "stamps" to prevent forgeries. Users can protect themselves against unauthorized use of their cards via a password system or by memorizing part of the key.

The differences between private-key, public-key, and identity-based cryptosystems are summarized in Fig. 1. In all these schemes, the mes-

sage  $m$  is encrypted under key  $k_e$ , transmitted as ciphertext  $c$  through the exposed channel, and decrypted under key  $k_d$ . The choice of keys is based on a truly random seed  $k$ . In private-key schemes,  $k_e = k_d = k$ , and the separate key channel (which is usually a courier) must preserve both the secrecy and the authenticity of the key. In public-key schemes, the encryption and decryption keys are derived from  $k$  via two different functions  $k_e = fe(k)$  and  $k_d = fd(k)$ , and the separate key channel (which is usually a directory) must preserve only the authenticity of the key. In identity-based schemes, the encryption key is the user's identity  $k_e = i$ , and the decryption key is derived from  $i$  and  $k$  via  $k_d = f(i, k)$ . The separate key channel between the users is completely eliminated, and is replaced by a single interaction with the key generation center when the recipient first joins the network.

Public-key and identity-based signature schemes are mirror images of the corresponding cryptosystems, as depicted in Fig. 2. The message  $m$  is signed with the signature generation key  $k_g$ , transmitted along with its signature  $s$  and sender identity  $i$ , and verified with the signature verification key  $k_v$ . The rest of the diagram should be self-evident.

## THE IMPLEMENTATION

To implement the idea described in the previous section, we need a public-key scheme with two additional properties: (a) When the seed  $k$  is known, secret keys can be easily computed for a non-negligible fraction of the possible public keys. (b) The problem of computing the seed  $k$  from specific public/secret key pairs generated with this  $k$  is intractable.

Unfortunately, the RSA scheme cannot be used in a way that satisfies these conditions simultaneously: (a) If the modulus  $n$  is a pseudo-random function of the user's identity, even the key generation center cannot factor this  $n$  and cannot compute the decryption exponent  $d$  from the encryption exponent  $e$ . (b) If the modulus  $n$  is universal and the seed is its secret factorization, then anyone who knows an encryption exponent  $e$  and its corresponding decryption exponent  $d$  can compute the seed.

At this stage we have concrete implementation proposals only for identity-based signature schemes, but we conjecture that identity-based cryptosystems exist as well and we encourage the reader to look for such systems. This situation is reminiscent of the 1976 period, when public key cryptosystems were defined and their potential applications were

investigated even though the first concrete implementations were published only in 1978.

The signature scheme is based on the verification condition

$$s^e = i \cdot t^{f(t,m)} \pmod{n}$$

where

- $m$  is the message
- $s, t$  is the signature
- $i$  is the user's identity
- $n$  is the product of two large primes
- $e$  is a large prime which is relatively prime to  $\varphi(n)$
- $f$  is a one way function.

The parameters  $n, e$  and the function  $f$  are chosen by the key generation center, and all the users have the same  $n, e$  and the same algorithmic description of  $f$  stored in their smart cards. These values can be made public, but the factorization of  $n$  should be known only to the key generation center. The only difference between users is the value of  $i$ , and the secret key which corresponds to  $i$  is the (unique) number  $g$  such that

$$g^e = i \pmod{n}.$$

This  $g$  can be easily computed by the key generation center, but if the RSA scheme is secure no one else can extract  $e$ -th roots mod  $n$ .

Each message  $m$  has a large number of possible  $(s, t)$  signatures, but their density is so low that a random search is extremely unlikely to discover any one of them. Any attempt to set one of  $(s, t)$  to a random value and solve for the other variable requires the extraction of modular roots, which is believed to be an exceedingly difficult computational task. However, when  $g$  is known, there is a very simple way to generate any number of signatures of any message even when the factorization of  $n$  is unknown.

To sign the message  $m$ , the user chooses a random number  $r$  and computes

$$t = r^e \pmod{n}.$$

The verification condition can be rewritten as

$$s^e = g^e \cdot r^{ef(t,m)} \pmod{n}.$$

Since  $e$  is relatively prime to  $\varphi(n)$ , we can eliminate the common factor  $e$  from the exponents

$$s = g \cdot r^{f(t,m)} \pmod{n}$$

and thus  $s$  can be computed without any root extraction.

To prevent attacks based on multiplicative relationships between the identities (and thus also the  $g$  values) of different users, it is advisable to expand the string that describes the user's identity into a long pseudo-random string via a universal one way function, and use the expanded form as  $i$  in the verification condition. Since everyone in the network knows how to apply this function, the scheme retains its identity-based flavour even though the signature verification key is not strictly equal to the user's identity.

The security of the scheme depends on the inability of the cryptanalyst to isolate  $g$  by analysing a large number of valid signatures of messages of his choice. If the gcd of  $f$  and  $e$  is  $c \neq 1$ , it is possible to extract the  $c$ -th root of  $i$  by manipulating the verification condition, and thus it is essential to make  $e$  a sufficiently large prime and  $f$  a sufficiently strong one way function so that this case will never arise in practice. The value of  $r$  should never be reused or revealed, since  $g$  is protected in any concrete signature by its randomness and secrecy.

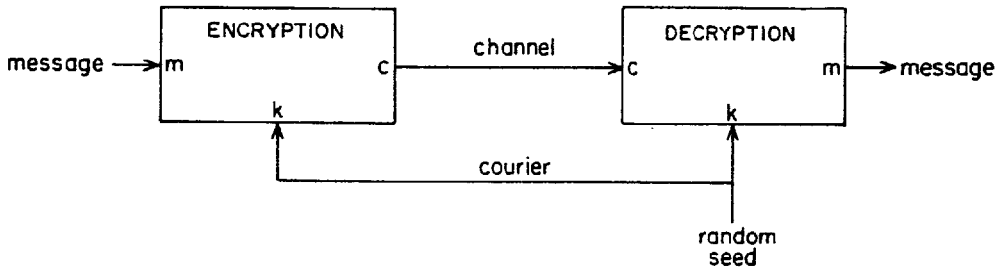
The variants of the verification condition in which one of the two occurrences of  $t$  is eliminated (e.g., by replacing it with a constant) are insecure. It is thus important to use a one way function that mixes  $t$  and  $m$  thoroughly (preferably via non-arithmetic and non-invertible operations) and which has a large range of possible values.

We believe that with a proper choice of parameters this scheme can be made very secure, but we cannot prove that breaking it is equivalent to solving some well known computational problem. Its main purpose is didactic, to serve as the first existence proof for identity based schemes. The Ong-Schnorr-Shamir signature scheme (described elsewhere in these proceedings) can also be used as an identity-based scheme, but its security is still an open problem in light of Pollard's successful attacks against its earlier versions. As always, we do not recommend to use this scheme right away, before the cryptographic community had ample time to assess its security.

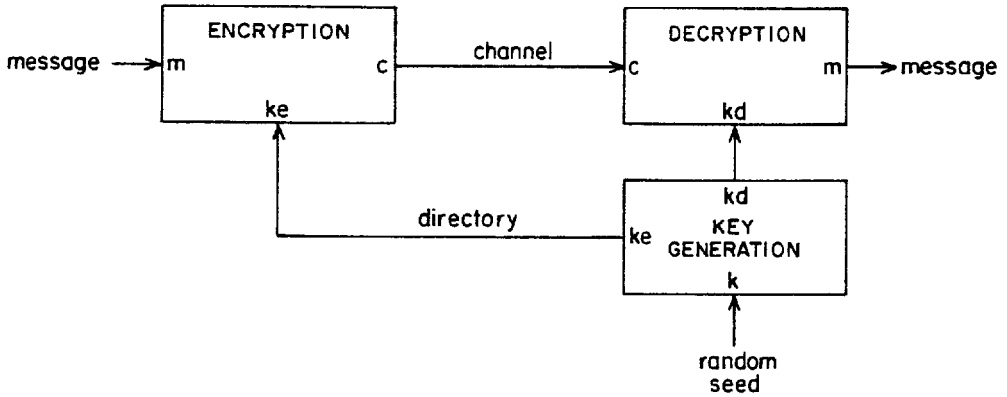
#### ACKNOWLEDGEMENTS

I would like to thank Amos Fiat, Heidroon Ong and Claus Schnorr for very helpful discussions.

## PRIVATE-KEY CRYPTOSYSTEM :



## PUBLIC-KEY CRYPTOSYSTEM :



## IDENTITY-BASED CRYPTOSYSTEM :

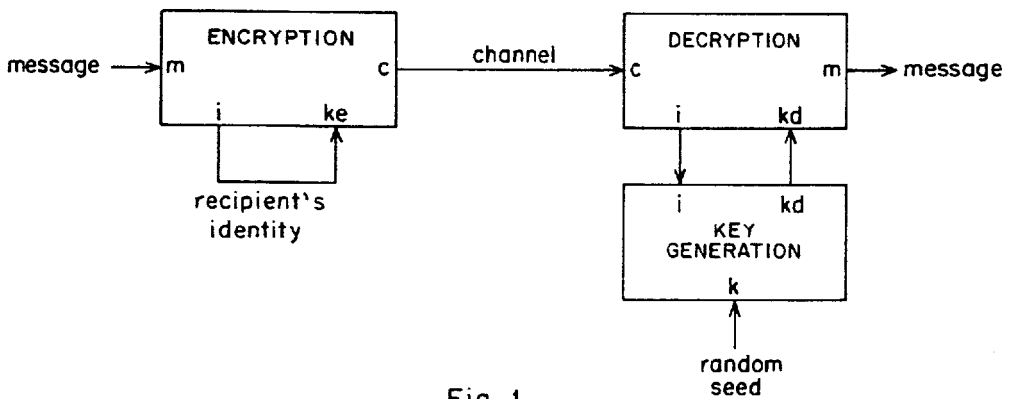
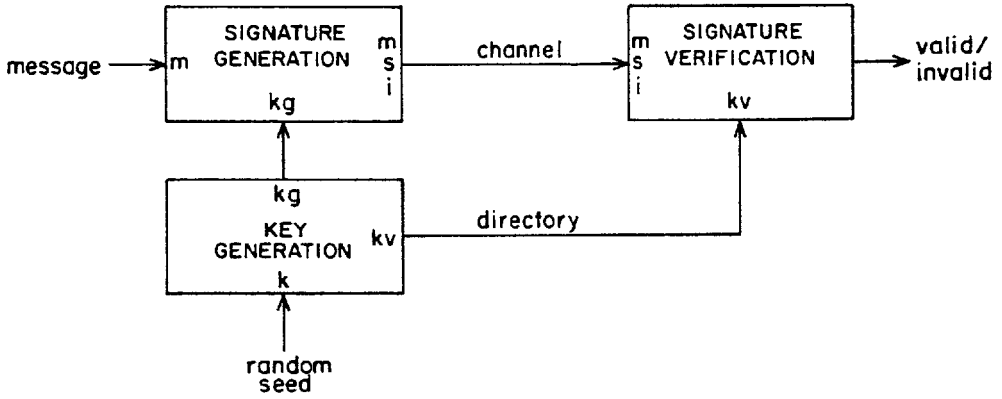


Fig. 1.

## PUBLIC-KEY SIGNATURE SCHEME :



## IDENTITY-BASED SIGNATURE SCHEME :

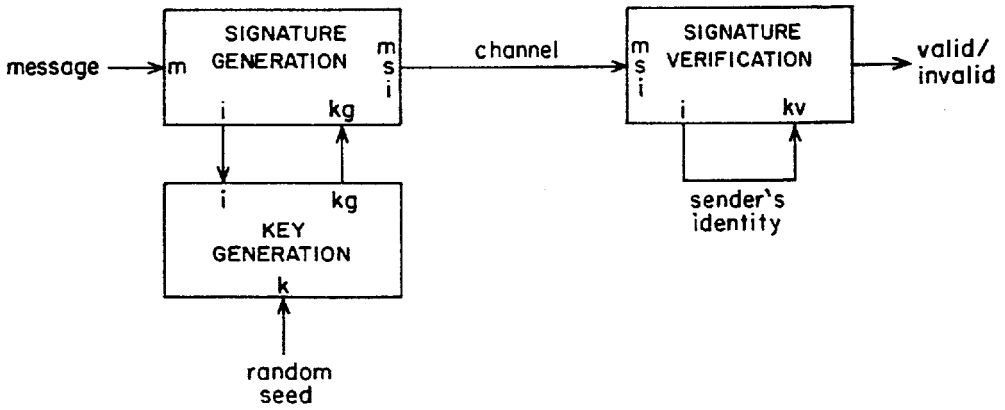


Fig. 2.

# A Knapsack Type Public Key Cryptosystem Based On Arithmetic in Finite Fields

(preliminary draft)

Benny Chor   Ronald L. Rivest

Laboratory for Computer Science  
Massachusetts Institute of Technology  
Cambridge, Massachusetts 02139

**Abstract**—We introduce a new knapsack type public key cryptosystem. The system is based on a novel application of arithmetic in finite fields, following a construction by Bose and Chowla. Appropriately choosing the parameters, we can control the density of the resulting knapsack. In particular, the density can be made high enough to foil “low density” attacks against our system. At the moment, we do not know of any attacks capable of “breaking” this system in a reasonable amount of time.

## 1. INTRODUCTION

In 1976, Diffie and Hellman [7] introduced the idea of public key cryptography, in which two different keys are used: one for encryption, and one for decryption. Each user keeps his decryption key secret, while making the encryption key public, so it can be used by everyone wishing to send messages to him. A few months later, the first two implementations of public key cryptosystems were discovered: The Merkle-Hellman scheme [13] and the Rivest-Shamir-Adelman scheme [17]. Some more PKC have been proposed since that time. Most of them can be put into two categories<sup>1</sup>:

- a. PKC based on hard number-theoretic problems ([17],[16],[8]).
- b. PKC related to the knapsack problem ([13],[2]).

While no efficient attacks against number theoretic PKC are known, some knapsack type PKC

---

<sup>†</sup> Research supported by NSF grant MCS-8006938. Part of this research was done while the first author was visiting Bell Laboratories, Murray Hill, NJ.

<sup>1</sup>with the exception of McEliece system [12], which is based on error correcting codes



were shown to be insecure. Most of those systems have a concealed "superincreasing" sequence. Shamir made the first successful attack on the basic Merkle-Hellman system (see [19]). Following his attack, other attacks against more complicated systems were proposed. The strongest of these seems to be the "low density" attack of Lagarias and Odlyzko [11]. The most interesting point about this last attack is that it does not make any assumption about how the system was constructed, and thus might be applicable to any knapsack type cryptosystem (unlike, say, Shamir's attack which relies heavily on the superincreasing underlying sequence). As a result of these attacks, knapsack type PKC which are either based on superincreasing sequences or have very low density seem to be vulnerable.

In this paper, we propose a new knapsack type PKC which has high density and a completely different basis. The underlying construction makes use of a result due to Bose and Chowla [1] about unique representation of sums in "dense" finite sequences. To implement this construction requires taking discrete logarithms in finite fields, for creating the encryption-decryption keys. Once this is done, encryption is very fast (linear time) and decryption is reasonably fast (comparable to RSA). Hence creating the keys is the hard part. While there are no polynomial time algorithms known for taking discrete logarithms, there are practical algorithms (most notably the ones due to Pohlig and Hellman [15] and Coppersmith [5]) in some special cases. We can demonstrate the existence of such special cases which would both yield reasonable size keys to foil the low density and exhaustive search attacks, and do so in reasonable amount of time (a few hours on a minicomputer, which is not too bad since keys are created only once per user). It should also be noticed that all known number theoretic PKC are at most as hard as factoring and hence are all reducible to the problem of taking discrete logarithms in composite moduli (see appendix 1). Should this discrete logarithm problem become tractable (thus rendering all "number-theoretic" PKC insecure), our system will become easier to create for even larger size knapsacks.

The remainder of this paper is organized as follows: In section 2 we discuss the knapsack problem and its use in cryptosystems. Section 3 describes Bose-Chowla theorem and its proof. In section 4 we give the details of our new cryptosystem. In section 5 the system performance is examined, and section 6 describes the actual parameters for implementing our PKC. Finally, some possible attacks against the new system are analyzed in section 7.

## 2. KNAPSACK-TYPE CRYPTOSYSTEMS

The 0 - 1 knapsack problem is the following NP-complete decision problem: Given a set  $A = \{a_i \mid 0 \leq i \leq n-1\}$  of non-negative integers and a non-negative integer  $S$ , is there an integer solution to  $\sum x_i a_i = S$  where all  $x_i$  are 0 or 1. A different variant of the problem is

to remove the 0 - 1 restriction on the  $x_i$  (but insisting they remain non-negative integers) and bounding their total weight  $\sum x_i \leq h$ .

Knapsack type public-key cryptosystems are based on the intractability of finding a solution to  $S = \sum x_i a_i$  even when a solution is known to exist. In such systems, each user publishes a set  $A$  of  $a_i$  and a bound  $h$ . A plaintext message consisting of an integer vector  $M = (x_0, x_1, \dots, x_{n-1})$  with weight  $\leq h$  is encrypted by setting

$$E(M) = \sum x_i a_i.$$

The knapsack elements  $a_i$  are chosen in such way that the equation is easily solved if certain secret *trapdoor* information is known. The exact nature of this information depends on the particular system in question. A general property of knapsack type PKC is that encryption is easy - all you have to do is to add.

### 3. BOSE-CHOWLA THEOREM

In 1936, Sidon raised the question of whether there exist "dense" sequences whose  $h$ -fold sums are unique. *Given  $n$  and  $h$ , non-negative integers, is there a sequence  $A = \{a_i \mid 0 \leq i \leq n-1\}$  of non-negative integers, such that all sums of exactly  $h$  elements (repetitions allowed) out of  $A$  are distinct?* It is easy to construct such sequences if the  $a_i$  are growing exponentially in  $n$ : For example, the sequence  $\{1, h, h^2, \dots, h^{n-1}\}$  has the above property (but does not work even for  $h+1$  element sums, since  $h^2 + h \cdot 1 = (h+1) \cdot h$ ). But can one construct such sequence with the  $a_i$  growing only polynomially fast in  $n$ ? Bose and Chowla [1] found a very elegant way of constructing such sequences with  $1 \leq a_i \leq n^h - 1$  (see [9, ch.2] for an overview of the subject). Here, we'll present a slightly modified version of Bose-Chowla theorem, which will fit well our cryptographic application.

**Bose-Chowla Theorem** *Let  $p$  be a prime,  $h \geq 2$  an integer. Then there exists a sequence  $A = \{a_i \mid 0 \leq i \leq p-1\}$  of integers such that*

1.  $1 \leq a_i \leq p^h - 1 \quad (i = 0, 1, \dots, p-1)$ .
2. *If  $(x_0, x_1, \dots, x_{p-1})$  and  $(y_0, y_1, \dots, y_{p-1})$  are two distinct vectors with non-negative integral coordinates and  $\sum_{i=0}^{p-1} x_i, \sum_{i=0}^{p-1} y_i \leq h$ , then  $\sum_{i=0}^{p-1} x_i a_i \neq \sum_{i=0}^{p-1} y_i a_i$ .*

*Proof:* The construction takes place in  $GF(p)$  and its  $h$ -degree extension,  $GF(p^h)$ . Let  $t \in GF(p^h)$  be algebraic of degree  $h$  over  $GF(p)$  ( i.e. the minimal polynomial in  $GF(p)[x]$  having  $t$  as its root is of degree  $h$  ). Let  $g$  be a multiplicative generator ( primitive element ) of  $GF(p^h)$ . Look at an additive shift by  $t$  of the base field,  $GF(p)$ , namely at the set  $t + GF(p)\{t + i \mid i = 0, 1, \dots, p-1\} \subseteq GF(p^h)$ .

Let  $a_i = \log_g(t + i)$  ( $i = 0, 1, \dots, p-1$ ) the logarithm of  $t + i$  to the base  $g$  in  $GF(p^h)$ . Then the  $a_i$  are all in the interval  $[1, p^h - 1]$  and they satisfy the distinctness of  $h$ -fold sums: For suppose there are two vectors  $\vec{x}, \vec{y}$  with

$$(x_0, x_1, \dots, x_{p-1}) \neq (y_0, y_1, \dots, y_{p-1}),$$

$$\sum_{i=0}^{p-1} x_i, \sum_{i=0}^{p-1} y_i \leq h, \quad \text{and} \quad \sum_{i=0}^{p-1} x_i a_i = \sum_{i=0}^{p-1} y_i a_i.$$

$$\text{Then also} \quad g^{\sum_{i=0}^{p-1} x_i a_i} = g^{\sum_{i=0}^{p-1} y_i a_i}$$

$$\text{and so} \quad \prod_{i=0}^{p-1} (g^{a_i})^{x_i} = \prod_{i=0}^{p-1} (g^{a_i})^{y_i}.$$

Since  $g^{a_i} = t + i$ , we get

$$(t + i_1)^{x_1} (t + i_2)^{x_2} \dots (t + i_l)^{x_l} = (t + j_1)^{y_1} (t + j_2)^{y_2} \dots (t + j_k)^{y_k},$$

where  $\{i_1, i_2, \dots, i_l\}$  and  $\{j_1, j_2, \dots, j_k\}$  are two different non-empty sets of elements from  $\{0, 1, \dots, p-1\}$ , with at most  $h$  elements each. Therefore, both sides of the last equation are monic distinct polynomials of degree  $\leq h$  with coefficients in  $GF(p)$ , so we can subtract them and get:

$t$  is a root of a non-zero polynomial, with coefficients in  $GF(p)$ , of degree  $\leq h-1$ .

This contradicts the fact that  $t$  is algebraic of degree  $h$  over  $GF(p)$ . ■

Remarks:

1. From the above proof it is clear that  $l$  sums ( $l \leq h$ ) of  $A$  are distinct not only over  $\mathbb{Z}$ , but also modulo  $p^h - 1$ .
2. The requirement " $p$  is a prime" can be replaced by " $p$  is a prime power" with no change in the claim or its proof.

#### 4. THE NEW CRYPTOSYSTEM

In this section we describe how the new cryptosystem is created and used. We start with an informal (and slightly simplified) description. Next, a step-by-step recipe for generating the cryptosystem, encrypting messages and decrypting cyphertexts is given.

The first step is to pick  $p$  and  $h$  such that  $GF(p^h)$  is amenable for discrete logarithm computations. We leave  $p$  and  $h$  as unspecified parameters in this section, and elaborate more on their exact choice in section 7 (the approximate magnitudes will be  $p \approx 200$ ,  $h \approx 25$ ). Once  $p$  and  $h$  are chosen, we pick  $t \in GF(p^h)$  of algebraic degree  $h$  over the base field, and a primitive element

$g \in GF(p^h)$  (both  $t$  and  $g$  are picked at random from the many possible candidates). Following Bose and Chowla, logarithms (to base  $g$ ) of the  $p$  elements in  $GF(p) + t$  are computed. These  $p$  integers are then scrambled, using a randomly chosen permutation. The scrambled integers are published. Together with  $p$  and  $h$ , they constitute the public key.

In order to encrypt a binary message of length  $p$  and weight  $h$ , a user adds the knapsack elements with 1 in the corresponding message location, and sends the sum. Section 6 deals with the question of transforming "regular", unconstrained binary strings to those of the above form.

When the legitimate receiver gets a sum, he first raises the generator  $g$  to it, and expresses the result as a degree  $h$  polynomial in  $t$  over  $GF(p)$ . The  $h$  roots of this polynomial are found by successive substitutions. Applying the inverse of the original permutation, the indices of the plaintext having the bit 1 are recovered.

#### a. System Generation

1. Let  $p$  be a prime power,  $h \leq p$  an integer such that discrete logarithms in  $GF(p^h)$  can be efficiently computed.
2. Pick  $t \in GF(p^h)$  -  $t$  algebraic of degree  $h$  over  $GF(p)$  at random. This will be done by finding  $f(t)$ , a random irreducible monic polynomial of degree  $h$  in  $GF(p)[t]$ , and representing  $GF(p^h)$  arithmetic by  $GF(p)[t]/\langle f(t) \rangle$  (where  $\langle f(t) \rangle$  is the ideal generated by  $f(t)$ ).
3. Pick  $g \in GF(p^h)$ ,  $g$  a multiplicative generator of  $GF(p^h)$  at random.
4. Construction following Bose-Chowla theorem: Compute  $a_i = \log_g(t+i)$  for  $i = 0, 1, 2, \dots, p-1$ .
5. Scramble the  $a_i$ 's: Let  $\pi : \{0, 1, \dots, p-1\} \rightarrow \{0, 1, \dots, p-1\}$  be a randomly chosen permutation. Set  $b_i = a_{\pi(i)}$ .
6. Add some noise: Pick  $0 \leq d \leq p^h - 2$  at random. Set  $c_i = b_i + c$ .
7. Public key - to be published:  $c_0, c_1, \dots, c_{p-1}; p, h$ .
8. Private key - to be kept secret:  $t, g, \pi, d$ .

Note: Every user will have the same  $p$  and  $h$ . The probability of collisions (two users having the same keys) is negligible.

#### b. Encryption

To encrypt a binary message  $M$  of length  $p$  and weight (number of 1's) *exactly*  $h$ , add the  $c_i$ 's whose corresponding bit is 1. Send

$$E(M) = c_{i_1} + c_{i_2} + \dots + c_{i_h} \pmod{p^h - 1}.$$

#### c. Decryption

0. Let  $r(t) = t^h \bmod f(t)$ , a polynomial of degree  $\leq h-1$  (computed once at system generation).
1. Given  $s = E(M)$ , compute  $s' = s - hd \bmod p^h - 1$ .
2. Compute  $p(t) = g^{s'} \bmod f(t)$ , a polynomial of degree  $h-1$  in the formal variable  $t$ .
3. Add  $t^h - r(t)$  to  $p(t)$  to get  $d(t) = t^h + p(t) - r(t)$ , a polynomial of degree  $h$  in  $GF(p)[t]$ .
4. We now have

$$d(t) = (t + i_1) \cdot (t + i_2) \dots (t + i_h)$$

namely  $d(t)$  factors to *linear terms* over  $GF(p)$ . By successive substitutions, we find the  $h$  roots  $i_j$ 's (at most  $p$  substitutions needed). Apply  $\pi^{-1}$  to recover the coordinates of the original  $M$  having the bit 1.

## 5. SYSTEM PERFORMANCE: TIME, SPACE AND INFORMATION RATE

In this section we analyze three basic parameters of the cryptosystem: The time needed for encrypting and decrypting a message, the size of the keys, and the information rate in terms of cleartext bits per ciphertext bits.

Given a binary message length  $p$  and weight  $h$ , encrypting it amounts to adding  $h$  integers  $c_i$ , each smaller than  $p^h$ . The run time for decryption is much longer. It is dominated by the modular exponentiation: To raise a polynomial  $g$  to a power in the range  $[1, p^h - 1]$  takes at most  $2h \log p$  modular multiplications. The modulus is  $f(t)$ , a polynomial of degree  $h$ , with coefficients in  $GF(p)$ . Using the naive polynomial multiplication algorithm,  $2h^2$  operations (in  $GF(p)$ ) per modular multiplication will suffice. So overall,  $4h^3 \log p$  operations in  $GF(p)$  are required. For the proposed parameters  $p \approx 200$ ,  $h \approx 25$  this gives about 500,000  $GF(p)$  operations, and compares favorably with RSA encryption-decryption time.

The size of the keys, and especially of the public key, is an important factor in the design of any public key system. In such system, a directory containing all public keys should be maintained such that each entry is easily accessible by every user. In our system, the size of the public key is that of  $p$  numbers, each in the range  $[1, p^h - 1]$ . In terms of bits, this is  $p \log_2 p^h = ph \log_2 p$  bits. For  $p \approx 200$ ,  $h \approx 25$ , the key takes less than 40,000 bits. While this number is about 35 times larger than the currently proposed size for the RSA public key (600 bits for the modulus and 600 for the exponent), it is still within practical bounds.

The information rate  $R$  of a block code is defined as  $R = \frac{\log_2 |M|}{N}$ , where  $|M|$  is the size of the message space, and  $N$  is the number of bits in a ciphertext. Letting  $M$  range over all binary vectors of length  $p$  and weight  $h$ ,  $|M| = \binom{p}{h}$ .  $N = \log_2 p^h$ , so the information rate is

$$R = \frac{\log \binom{p}{h}}{\log p^h} .$$

For the proposed parameters  $p = 197$ ,  $h = 24$ ,  $R = 0.556$  (data expansion 1.798).

## 6. PROPOSED PARAMETERS

As mentioned before, the main obstacle in implementing our cryptosystem is the computation of discrete logarithms in large finite fields  $GF(p^h)$ . This computational problem is considered quite hard in general. However, the algorithms of Coppersmith [5] and Pohlig and Hellman [15] work well in practice for some special cases. Coppersmith algorithm is appropriate for fields of small characteristic, and performs best in characteristic 2. Letting  $p^h = 2^n$ , the run time of the algorithm is  $O\left(\sqrt[3]{n \log^2 n}\right)$ . For  $n \leq 200$ , implementation of Coppersmith algorithm will terminate in a few hours on a mainframe computer. Pohlig-Hellman algorithm works for any characteristic, provided  $p^h - 1$  has only small prime factors. It turns out that Pohlig-Hellman algorithm is preferable for our specific application, due to two properties: The simplicity of the algorithm, and the nice factorization of several numbers  $p^h - 1$  of appropriate magnitude.

The Pohlig-Hellman algorithm has a  $T \cdot S$  (time-space) complexity proportional to the largest factor of  $p^h - 1$ . While in general numbers whose order of magnitude is  $\approx 200^{25}$  do not have 'small' largest factors (the expected size of the largest factor of a number  $m$  is about  $m^{0.6}$ ), things are much better when the number has the form  $x^h - 1$ , since we can first factor this expression as a polynomial in  $x$ , and then factor each term as a number after substituting  $x \leftarrow p$ .  $h$ 's with "good" factorization are especially effective. For example,  $x^{24} - 1$  has the factors  $x^8 - x^4 + 1$ ,  $x^4 - x^2 + 1$ ,  $x^4 + 1$ , and other terms of degree not exceeding 2. Substituting  $p = 197$ , the largest prime factor of  $197^{24} - 1$  is  $10,316,017 \approx 10^7$ . The square root of this is  $3 \cdot 10^3$ , so Pohlig-Hellman algorithm can easily be implemented on a minicomputer within a few CPU hours for all the 197 logarithms.

Other possible values are (the last two values are from [4]):

- $p = 211$ ,  $h = 24$  (largest prime factor of  $211^{24} - 1$  is  $216,330,241 \approx 2 \cdot 10^8$ )
- $p = 256 = 2^8$ ,  $h = 25$  (largest prime factor of  $2^{200} - 1$  is  $3,173,389,601 \approx 3 \cdot 10^9$ ). This candidate has the advantage of using binary arithmetic for decryption calculations.
- $p = 243 = 3^5$ ,  $h = 24$  (largest prime factor of  $3^{120} - 1$  is  $47,763,361 \approx 5 \cdot 10^7$ ).

## 7. POSSIBLE ATTACKS

In this section we examine some possible attacks on the cryptosystem. We start with attacks where part of the secret key is known to the cryptanalyst and he is trying to reconstruct the rest of it. We proceed by considering low density and brute force attacks with no prior secret information, where the goal is not to reconstruct the secret key but rather to decipher a given ciphertext.

a. Known  $g$  and  $d$ .

Let  $t' = g^{c_0}$ , then  $t - t' \in GF(p)$ , so the sets  $\{t + i | i \in GF(p)\}$  and  $\{t' + i | i \in GF(p)\}$  are identical. Therefore, by using  $t'$ , the cryptanalyst can determine  $\pi$  and has all needed information for decryption.

b. Known  $t$  and  $d$ .

Pick arbitrary generator  $g'$ . Compute  $a'_i = \log_{g'}(t + i)$ . As sets, we have

$$\{a_0, a_1, \dots, a_{p-1}\} = L\{a'_0, a'_1, \dots, a'_{p-1}\}$$

where equality is modulo  $p^h - 1$  and  $L, p^h - 1$  are relatively prime,  $L$  satisfying  $g = g'^L$ . Once  $L$  is recovered, we are done, for then  $g = g'^L$ , and we can reconstruct  $\pi$  and have all the pieces of the private key.

If one of the  $a'_i$  ( $a'_0$ , say) is relatively prime<sup>1</sup> to  $p^h - 1$ , then  $L$  is one of  $a_j a'^{-1}_0 \pmod{p^h - 1}$ , for some  $0 \leq j \leq p - 1$ . Otherwise, the cryptanalyst can compute  $L$  modulo each of the prime power factors of  $p^h - 1$  (which are all small and therefore easy to find, by the choice of  $p$  and  $h$ ), and then combine them together using the Chinese remainder theorem.

c. Known permutation  $\pi$  and  $d$  (attack due to Andrew Odlyzko).

Since the knapsack is dense, there are small integral coefficients  $x_i$  (some of which may be negative) such that

$$\sum_{i=0}^{p-1} x_i a_i = 0$$

(for details see [14]). Furthermore, the LLL algorithm can find these  $x_i$ 's. The last equality implies

$$g^{\sum_{i=0}^{p-1} x_i a_i} = 1$$

i.e.

$$\prod_{i=0}^{p-1} (t + i)^{x_i} = 1.$$

The left hand side of the last equality is a rational function of  $t$ , and  $g$  (which is still unknown) is not a part of it. If  $m_1 = |\sum x_i^+|$  ( $m_2 = |\sum x_i^-|$ ) denotes the sum of positive (negative)  $x_i$ 's, and  $m = \max(m_1, m_2)$ , then we get a polynomial equation of degree  $m - 1$  in  $t$ , with coefficients from  $GF(p)$ . All roots (in  $GF(p^h)$ ) of this polynomial can be found using a fast probabilistic algorithm.  $t$  is necessarily one of these roots, so attack (b) can now be used.

<sup>1</sup>this means that one of the  $t + i$  is itself a multiplicative generator of  $GF(p^h)$ , and will happen with high probability.

Remark: If  $\pi$  is not known, this attack does not seem to work since, even though the  $x_i$  can be found, they give rise to an 'unknown' polynomial. If  $m_1 + m_2$  is very small then one can try all  $\binom{p}{m_1+m_2}$  possibilities even without knowing  $\pi$ . However, with  $\pi$  unknown and  $m_1 + m_2$  exceeding 10, this approach becomes infeasible.

#### d. Low density attacks

Brickel [3] and independently Lagarias and Odlyzko [11] introduced "general purpose" algorithms which can be expected to recover successfully the added elements of any "low density" knapsack system. In this subsection we briefly describe the second method, and examine its success when applied to our system.

The *density*  $d(A)$  of a knapsack system  $A = \{a_i | 0 \leq i \leq p-1\}$ , is defined to be

$$d(A) = \frac{p}{\log_2(\max a_i)}.$$

Given a knapsack system  $A = \{a_i | 0 \leq i \leq p-1\}$  and a sum instance (ciphertext)  $S = \sum_{i=0}^{p-1} x_i a_i$ , Lagarias and Odlyzko construct a  $p+1$  dimensional lattice. The lattice construction uses the  $p$  knapsack elements and the given ciphertext. A certain vector in this lattice (which we call here the *special vector*) is defined. This vector corresponds to the solution of the given ciphertext (yields the coefficients  $x_i$  in the sum), and the goal of the cryptanalyst is to find it. Lagarias and Odlyzko have shown that if  $d(A)$  is low, this special vector is the shortest one in the lattice.

Using the last observation, what Lagarias and Odlyzko are trying to do is to find the shortest vector in the lattice. The tool they use is the basis reduction algorithm of Lenstra, Lenstra and Lovasz. While this algorithm usually succeeds if the shortest vector in the lattice is much shorter than all other vectors, it does not do so well if the shortest vector is relatively close in length to other vectors.

In our specific case, the knapsack has high density. The length (square of Euclidean norm) of the specific vector will not be much shorter than the length of many other vectors (24 vs. 40 for  $p = 197$ ,  $h = 24$ ), and so the LLL algorithm cannot be expected to find it. Experiments, done by Andrew Odlyzko, on a smaller knapsack created by us ( $p = 103$ ,  $h = 12$ , a system with density 1.271, where the calculations imply that all vectors other than the specific one have length at least 17, but the LLL algorithm did not find the specific vector even when its length was only 5), support this claim. So, for Lagarias-Odlyzko attack to be successful against our system, it must use a better shortest vector algorithm. Currently, the best (exact) shortest vector algorithm known is the one of Kanaan [10], and its performance is no better, in our application, than the brute force attack sketched in the next subsection.



We wish to remark that it is possible to make the specific vector which solves a given ciphertext *longer*, by reducing the information rate of the system, without changing its density (details are left to the full paper). In this case, no shortest vector algorithm will find this vector. However, with the current state of shortest vector algorithm, it looks like such modification to the cryptosystem is not required.

#### e. Brute force attacks.

The most efficient method we know of for solving knapsack instances with  $h$  out of  $p$  items, given a specific ciphertext, is the following: There are  $\binom{p}{h}$  ways of choosing  $h$  out of  $p$  elements. Take a random subset  $S$  containing  $p/2$  elements. The probability that a given sum contains exactly  $h/2$  out of these  $p/2$  elements is

$$\frac{\binom{p/2}{h/2}^2}{\binom{p}{h}} \approx \frac{1}{\sqrt{h}}.$$

Assuming that this is indeed the case, we generate all  $h/2$  sums of  $S$  and of its complement, and sort them. The goal is to find a pair of sums from the two lists whose sum matches the desired target. This can be achieved by keeping two pointers to the two lists, and marching linearly through each (one in increasing order, and the other in decreasing order). If the two lists are exhausted but no matching sum was found, then another random  $S$  is tried. The run time per one choice of  $S$  is dominated by sorting all  $h/2$  sums of both  $S$  and its complement. This will require  $2 \cdot \binom{p/2}{h/2} \ln \binom{p/2}{h/2}$  operations. On the average, about  $\sqrt{h}$  choices of  $S$  have to be made. The overall expected running time will thus be

$$2 \cdot \binom{p/2}{h/2} \ln \binom{p/2}{h/2} \frac{\binom{p}{h}}{\binom{p/2}{h/2}^2} = \frac{2 \binom{p}{h} \ln \binom{p}{h}}{\binom{p/2}{h/2}}.$$

For  $p = 197$ ,  $h = 24$  the expected number of operations is  $3.466 \cdot 10^{17} > 2^{58}$ , so such brute force attack is totally impractical<sup>1</sup>.

Even though none of these attacks seems to produce a threat to the system security, other attacks might be successful. We urge the reader to examine our proposal for as yet undiscovered weaknesses.

## ACKNOWLEDGEMENTS

We wish to thank Don Coppersmith, Oded Goldreich, Jeff Lagarias and Andrew Odlyzko for many discussions concerning the system and possible attacks on it. Andrew's assistance in a

<sup>1</sup>the knapsack algorithm of Schroeppel and Shamir [18] might be used here as well for space efficiency. However, its run time behavior is no better than the above algorithm.

first implementation of the system, and later in testing the low density attack against it, was especially helpful. Oded was kind enough to decipher a few earlier versions of this manuscript, and his comments made it much clearer. Finally, thanks to Don Coppersmith and Victor Miller for acquainting us with [4] and [6].

## REFERENCES

- [1] Bose, R.C. and S. Chowla, "Theorems in the additive theory of numbers", *Comment. Math. Helvet.*, vol. 37, pp. 141-147, 1962.
- [2] Brickell, E.F., "A new knapsack based cryptosystem", Presented in Crypto83.
- [3] Brickell, E.F., "Are most low density knapsacks solvable in polynomial time?", *Proceedings of the Fourteenth Southeastern Conference on Combinatorics, Graph Theory and Computing*, 1983.
- [4] Brillhart, J., D.H. Lehmer, J.L. Selfridge, B. Tuckerman and S.S. Wagstaff, Jr., *Factorization of  $b^n \pm 1$* , in *Contemporary Mathematics*, vol. 22, AMS, Providence, 1983.
- [5] Coppersmith, D., "Fast Evaluation of Logarithms in Fields of Characteristic Two", to appear, *IEEE Trans. Inform. Theory*; extended abstract in *Proceedings of the Sixteenth Annual Symposium on Theory of Computing*, ACM, pp. 201-207, 1984.
- [6] Cover, T.M., "Enumerative Source Encoding", *IEEE Trans. Inform. Theory*, vol IT-19, pp. 73-77, 1973.
- [7] Diffie, W. and M. Hellman, "New directions in cryptography", *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644-654, 1976.
- [8] Goldwasser, S. and S. Micali, "Probabilistic Encryption", *Proceedings of the Fourteenth Annual Symposium on Theory of Computing*, ACM, pp. 365-377, 1982.
- [9] Halberstram, H. and K.F. Roth, *Sequences*, Springer-Verlag, New York, 1983.
- [10] Kannan, R., "Improved algorithms for integer programming and related lattice problems", *Proceedings of the Fifteenth Annual Symposium on Theory of Computing*, ACM, pp. 193-206, 1983.
- [11] Lagarias, J.C. and A.M. Odlyzko, "Solving low-density subset sum problems", *Proceedings of the Twenty-Fourth Annual Symposium on Foundations of Computer Science*, IEEE, pp. 1-10, 1983.
- [12] McEliece, R.J., "A public-key cryptosystem based on algebraic coding theory", *DSN Progress Report 42-44*, pp. 114-116, 1978.
- [13] Merkle, R.C. and M.E. Hellman, "Hiding information and signatures in trap-door knap-

- sacks", *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 525-530, 1978.
- [14] Odlyzko, M.O., "Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme", preprint, 1983.
  - [15] Pohlig, R.C. and M. Hellman, "An improved algorithm for computing logarithms over  $GF(p)$  and its cryptographic significance", *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106-110, 1978.
  - [16] Rabin, M.O., "Digitalized signatures and public-key functions as intractable as factorization", Technical report MIT/LCS/TR-212, MIT, 1979.
  - [17] Rivest, R.L., A. Shamir and L. Adelman, "On digital signatures and public key cryptosystems", *Commun. ACM*, vol. 21, pp. 120-126, 1978.
  - [19] Shamir, A., "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem", *Proceedings of the Twenty-Third Annual Symposium on Foundations of Computer Science*, IEEE, pp. 145-152, 1982.
  - [18] Schroepel, R. and A. Shamir, "A  $T = O(2^{n/2})$ ,  $S = O(2^{n/4})$  algorithm for certain NP-complete problems", *SIAM J. Comput.*, vol. 10, No. 3, pp. 456-464, 1981.

## Appendix 1: Discrete logarithms and factorization.

We'll show here how the problem of factoring "paired primes"  $n = p \cdot q$  ( $p, q$  primes) is polynomially reducible to that of finding indices in  $Z_n$ . Let  $a \in Z_n^*$ . Since  $a^{\varphi(n)} = 1 \pmod{n}$ , we have

$$a^n = a^{n-\varphi(n)} = a^{pq-(p-1)(q-1)} = a^{p+q-1} \pmod{n}.$$

The index of  $a^{p+q-1}$  to base  $a$  is a divisor of  $p+q-1$ , most likely  $p+q-1$  itself. Hence a discrete logarithm subroutine will output  $p+q-1$  when given  $a^n \pmod{n}$  as input. Having  $n = p \cdot q$  and  $p+q-1$ ,  $p$  and  $q$  can easily be determined.

SOME PUBLIC-KEY CRYPTO-FUNCTIONS  
AS INTRACTABLE AS FACTORIZATION

H. C. Williams  
Dept. of Computer Science  
University of Manitoba  
Winnipeg, Manitoba  
CANADA R3T 2N2

(Extended Abstract)

## 1. INTRODUCTION

It is well known that if one can factor the modulus  $R = pq$  ( $p, q$  distinct large primes) of the RSA cryptosystem [4], then the system can be broken. However, it is not known whether the problem of breaking an RSA cryptosystem is equivalent in difficulty to factoring  $R$ . Rabin [3] has given a public-key encryption method which is as difficult to break as it is to factor  $R$ , but the decryption process produces four possible candidates for the correct message and only one of these is the correct one. If the actual message being transmitted has little or no internal redundancy (e.g., a cryptographic key) there is no way for the sender to allow the recipient to identify the correct message being transmitted. Also, in [1] Lipton has pointed out some other weaknesses in this scheme when it is used as a cryptosystem. Indeed, Rabin only advocated its use as a signature system.

In [5], Williams described a modification of the RSA technique for which it can be shown that breaking the system is equivalent in difficulty to factoring a special form of  $R$  (viz.  $R = pq$ , where  $p \equiv 3 \pmod{8}$ ,  $q \equiv 7 \pmod{8}$ ). In this system the decryption procedure yields only one message, the one being transmitted. Also, if one uses a large value of  $e$ , Lipton's attacks will fail. The difficulty with this system is the fact that  $R$  must be of the above-mentioned special form. In this paper we describe a new method of public-key encryption, which has all of the advantages of that given in [5] but for which  $R$  can be a product of two arbitrary primes.

## 2. SOME PROPERTIES OF THE FUNCTIONS $X_n, Y_n$ .

Instead of raising an integer to a power modulo  $R$ , as is done in the RSA case, we raise a number  $\alpha$  of the form  $W_1 + \sqrt{C} W_2$  to a power modulo  $R$ . Here  $W_1, W_2, C$  are integers. We effect this by means of

the functions  $X_n$  and  $Y_n$ , defined by

$$X_n(W_1, W_2) = (\alpha^n + \bar{\alpha}^n)/2$$

$$Y_n(W_1, W_2) = W_2(\alpha^n - \bar{\alpha}^n)/(\alpha - \bar{\alpha}).$$

Here  $\bar{\alpha} = W_1 - \sqrt{C} W_2$  and we easily see that

$$\alpha^n = X_n(W_1, W_2) + \sqrt{C} Y_n(W_1, W_2).$$

Both functions  $X_n$  and  $Y_n$  can be computed modulo  $R$  in  $O(\log n)$  multiplication and division operations. (See, for example, Lehmer [2]).

We require the following

Theorem. Suppose  $p$  and  $q$  are odd primes and

- i)  $W_1^2 - C W_2^2 \equiv 1 \pmod{R}$ ;
- ii) The Legendre symbols  $\epsilon_p = (C/p) \equiv -p \pmod{4}$  and  $\epsilon_q = (C/q) \equiv -q \pmod{4}$ ;
- iii)  $\gcd(CW_2, R) = 1$ ;
- iv) The Jacobi symbol  $(2(W_1 + 1)/R) = 1$ ;
- v)  $ed \equiv (w + 1)/2 \pmod{w}$ ,

where

$$w = (p - \epsilon_p)(q - \epsilon_q)/4. \quad (1)$$

Then we must have

$$X_{2ed}(W_1, W_2) \equiv sW_1$$

$$Y_{2ed}(W_1, W_2) \equiv sW_2 \pmod{R},$$

where  $|s| = 1$ .  $\square$

### 3. ENCRYPTION AND DECRYPTION.

To produce his encryption key a designer of our cryptosystem selects two large primes  $p$  and  $q$  and forms their product  $R$ . He then selects, by trial, some integer  $C$  such that

$$(C/p) \equiv -p, (C/q) \equiv -q \pmod{4}.$$

Again, by trial, he finds some integer  $A$  such that the Jacobi symbol

$$(A^2 - C/R) = -1.$$

Finally, for a randomly selected  $e$  such that  $\gcd(e, w) = 1$  ( $w$  given

by (1)), he solves

$$ed \equiv (w + 1)/2 \pmod{w}$$

for  $d$ .

The public encryption key is now  $\{R, e, C, A\}$  and the secret decryption key is  $d$ . Since  $C$  and  $A$  are usually small, the encryption key does not require much more storage space than that required by the RSA scheme.

Let  $M$  be any message ( $0 < M < R$ ) to be encrypted. We put  $b_1 = 0$  when the Jacobi symbol  $(M^2 - C/R) = 1$  and we put  $b_1 = 1$  when  $(M^2 - C/R) = -1$ . Define

$$\gamma = \begin{cases} M + \sqrt{C} & \text{when } b_1 = 0 \\ (M + \sqrt{C})(A + \sqrt{C}) & \text{when } b_1 = 1. \end{cases}$$

We have  $(\gamma\bar{\gamma}/R) = 1$ .

Define  $T(M)$  and  $S(M)$  by

$$\alpha = \gamma/\bar{\gamma} \equiv T(M) + S(M)\sqrt{C} \pmod{R}.$$

(We say that  $(U + V\sqrt{C})/W \equiv T + S\sqrt{C} \pmod{R}$  when  $UW^{-1} \equiv T$  and  $VW^{-1} \equiv S \pmod{R}$ ). Note that

$$1 = \alpha\bar{\alpha} \equiv T(M)^2 - CS(M)^2.$$

Further,

$$2(T(M) + 1) \equiv (\gamma + \bar{\gamma})^2(\gamma\bar{\gamma})^{-1} \pmod{R};$$

thus

$$(2(T(M) + 1)/R) = 1.$$

If we put  $b_2 \equiv T(M) \pmod{2}$  ( $b_2 = 0, 1$ ), we can encrypt  $M$  by computing

$$E(M) \equiv X_e(T(M), S(M)) \{Y_e(T(M), S(M))\}^{-1} \pmod{R},$$

where  $0 < E(M) < R$ .

If  $E(M)$ , along with  $b_1$  and  $b_2$ , is sent to the designer of the cryptosystem, then he can compute

$$U \equiv X_{2e}(T(M), S(M)) \equiv (K^2 + C)(K^2 - C)^{-1}$$

$$V \equiv Y_{2e}(T(M), S(M)) \equiv 2K(K^2 - C)^{-1} \pmod{R},$$

where  $K = E(M)$ . Only he can now determine

$$X_d(U,V) \equiv X_{2ed}(T(M), S(M))$$

$$Y_d(U,V) \equiv Y_{2ed}(T(M), S(M)) \pmod{R}.$$

By the theorem we know that

$$X_d(U,V) \equiv sT(M), Y_d(U,V) \equiv sS(M) \pmod{R},$$

where  $|s| = 1$ . With the value of  $b_2$ , it is easy to find  $s$  and with  $b_1$ , it is easy to determine  $M$  from knowledge of both  $T(M)$  and  $S(M)$ .

#### 4. SECURITY.

Suppose that  $P$  is some algorithm which decrypts  $1/k$  of all ciphertexts produced by our cryptosystem. Find some  $Z$  such that

$$(Z^2 - C/R) = -1$$

and

$$\gcd(Z, R) = 1.$$

Compute

$$K \equiv X_e(T, S) Y_e(T, S)^{-1} \pmod{R},$$

where

$$T + S\sqrt{C} \equiv (Z + \sqrt{C}) / (Z - \sqrt{C}) \pmod{R}.$$

With probability  $1/k$ ,  $P$  will determine some  $M$  such that  $P(M) = K$ . Now for  $N$  one of  $M$  or  $C^{-1}M \pmod{R}$ , we have

$$(N^2 - C/R) = 1$$

and

$$Z(N^2 + C) \equiv N(Z^2 + C) \pmod{R};$$

hence,  $\gcd(N - Z, R) = p$  or  $q$  and  $R$  can be factored. Thus, anyone who can decrypt messages sent on this system can factor  $R$ .

We must emphasize, however, that since the method of proof utilized here is constructive, this system is susceptible to a known ciphertext attack. Thus, it must be used with certain protocols. Lipton's types of attack, when applied to this scheme, will also fail when  $e$  is large; indeed, even if one of the  $e$ 's is 1, his first attack will fail.

This scheme can also be used to produce signatures, but in this application  $e$  must not be 1.

The full details of the work summarized here will appear in a forthcoming issue of Cryptologia.

#### REFERENCES

- [1] R. A. Demillo et al. "On the Safety of Cryptosystems," Applied Cryptology, Cryptographic Protocols and Computer Security Models, AMS Short Course Lecture Notes, Vol. 29, Providence, 1983.
- [2] D. H. Lehmer, Computer technology applied to the theory of numbers, Studies in Number Theory, MAA Studies in Mathematics Vol. 6, 1969, pp. 117-151.
- [3] M. O. Rabin, "Digitized signatures and public-key functions as intractable as factorization," M.I.T. Lab. for Computer Science, Tech. Rep. LCS/TR212, 1979.
- [4] R. L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," Comm. ACM, 21 (1978), 120-126.
- [5] H. C. Williams, "A modification of the RSA public-key encryption procedure," IEEE Transactions on Information Theory, IT-26 (1980), 726-729.



# COMPUTING LOGARITHMS IN $GF(2^n)^*$

I.F. Blake<sup>1</sup>, R.C. Mullin<sup>2</sup>, S.A. Vanstone<sup>2</sup>

<sup>1</sup>Department of Electrical Engineering  
University of Waterloo  
Waterloo, Ontario, Canada  
N2L 3G1

<sup>2</sup>Department of Combinatorics and Optimization  
University of Waterloo  
Waterloo, Ontario, Canada  
N2L 3G1

## 1. The Problem.

Consider the finite field having  $q$  elements and denote it by  $GF(q)$ . Let  $\alpha$  be a generator for the nonzero elements of  $GF(q)$ . Hence, for any element  $b \neq 0$  there exists an integer  $x$ ,  $0 \leq x \leq q-2$ , such that  $b = \alpha^x$ . We call  $x$  the discrete logarithm of  $b$  to the base  $\alpha$  and we denote it by  $x = \log_\alpha b$  and more simply by  $\log b$  when the base is fixed for the discussion. The discrete logarithm problem is stated as follows:

Find a computationally feasible algorithm to compute  $\log_\alpha b$  for any  $b \in GF(q)$ ,  $b \neq 0$ .

Several cryptographic schemes have been proposed which base their security on the intractability of the discrete logarithm problem for large  $q$ .

In 1976 Diffie and Hellman [7] proposed the following public key passing scheme. Let  $A$  and  $B$  be two parties who wish to share a common key  $K$ . The finite field  $GF(q)$  and a generator  $\alpha$  are stored in a public file. Party  $A$  generates a random integer  $a$  and computes  $\alpha^a$ . Party  $B$  generates a random integer  $b$  and computes  $\alpha^b$ .  $A$  sends to  $B$  the field element  $\alpha^a$  and  $B$  sends to  $A$  the field element  $\alpha^b$ .  $A$  computes  $(\alpha^b)^a = \alpha^{ab}$  and  $B$  computes  $(\alpha^a)^b = \alpha^{ab}$ .  $A$  and  $B$  now share the common key  $K = \alpha^{ab}$ .

Recently, ElGamal [8] has proposed a public key cryptosystem and an authentication scheme which is based on discrete exponentiation. We describe only the public key system here.

Consider  $GF(q)$  generated by  $\alpha$ . The message space  $M$  will consist of all nonzero field elements. Party  $A$  generates a random integer  $a$  and stores  $\alpha^a$  in a public file. Suppose party  $B$  wishes to send a message  $m$  to  $A$ .  $B$  generates a random integer  $k$  and computes  $(\alpha^a)^k$  (since  $\alpha^a$  is made public this is possible),  $\alpha^k$  and  $m\alpha^{ak}$ .  $B$  sends  $A$  the pair  $(\alpha^k, \alpha^{ak}m)$ . Since  $A$  knows  $a$  he can compute  $(\alpha^k)^a = \alpha^{ak}$ , and, hence, obtain  $m$  from  $\alpha^{ak}m$ . This scheme and the Diffie-Hellman method appear to have the same degree of security. It remains an open problem as to whether or not the security of these systems is entirely dependent on comput-

\*Research supported in part by the Department of Communications under contract # 20ST 36001-4-0853.

ing discrete logarithms.

A number of special purpose algorithms for computing discrete logs have appeared in the literature (see, for example, [2], [9], [11], [14]). In this article we address only the most general methods currently available.

In the next three sections we describe three subexponential algorithms for computing logs. The algorithms of sections 3 and 4 are variants of the one presented in section 2. These algorithms are referred to as the index-calculus algorithms in an excellent and in-depth article on the subject by Odlyzko [13].

For the purposes of this paper we restrict our discussion to  $GF(2^n)$ . (The algorithms of sections 2 and 3 apply more generally.) We think of the elements in  $GF(2^n)$  as polynomials of degree at most  $n-1$  over  $GF(2)$  and multiplication is performed modulo some fixed irreducible polynomial of degree  $n$  over  $GF(2)$ . The examples cited in this paper refer to  $GF(2^{127})$  as it has been of some interest recently. (See, for example, [15], [17]).

## 2. Adleman Algorithm

The basic ideas involved in the following subexponential algorithm for computing discrete logarithms are due to Western and Miller [16]. Adleman [1] independently discovered the algorithm and partially analysed its computational complexity.

The algorithm consists of two parts. The first part requires the construction of a large database of logarithms. This database only needs to be constructed once for  $GF(2^n)$ . Part 2 of the algorithm computes individual logarithms.

### Part 1 (Database).

Find the logarithms of all irreducible polynomials of degree at most  $b$  where  $b$  is a fixed positive integer determined by  $GF(2^n)$ .

### Part 2.

To find the log of an element  $g(x) \in GF(2^n)$ ,  $g(x) \neq 0$ , generate a random integer  $a$  and compute  $h(x) = g(x)\alpha^a(x)$  where  $\alpha(x)$  generates  $GF(2^n)$ . Now, factor

$$h(x) = \prod_{i=1}^t p_i^{e_i}(x).$$

If each irreducible factor  $p_i(x)$  has  $\deg p_i(x) \leq b$  then

$$\log g(x) = \sum_{i=1}^t e_i \log p_i(x) - a$$

which can easily be evaluated by looking up  $\log p_i(x)$ ,  $1 \leq i \leq t$ , in the database. If not all  $p_i(x)$  have  $\deg p_i(x) \leq b$  then generate another random integer and repeat.

We define  $p(n, b)$  to be the probability that a randomly chosen polynomial of degree exactly  $n$  has all of its irreducible factors with degrees of at most  $b$ . If  $N(n, b)$  is the number of polynomials of degrees exactly  $n$  such that each has all of its irreducible factors with degree at most  $b$  then

$$p(n, b) = \frac{N(n, b)}{2^n}.$$

and the expected runtime of the second part of the algorithm should be approximately  $p(n, b)^{-1}$ . Odlyzko [1] shows that

$$p(n, b)^{-1} \cong \left( \frac{n}{b} \right)^{(1+O(1)) \frac{n}{b}}$$

and that the asymptotic running time of the entire algorithm is  $\exp(c_1(n \log n)^{1/4})$ .

Let  $S$  be the set of all irreducible polynomials of degree at most  $b$ . In order to find the logarithms of all elements in  $S$  we set up a system of  $|S|$  linear equations in  $|S|$  unknowns where the unknowns are the logarithms. We can find this system of equations by applying part 2 of the algorithm to each element of  $S$ . The resulting system must be solved over the integers modulo  $2^n - 1$ . The number of iterations of part 2 to produce the necessary equations is approximately  $|S|p(n, b)^{-1}$ . For  $GF(2^{127})$  this quantity is minimized by  $b=23$ . Since there are 766,150 irreducible polynomials of degree at most 23 then we require this many equations. A random polynomial of degree at most 126 will factor into the database with  $b=23$  with probability .000138. This means that to produce the desired set of equations will require about  $5,549 \times 10^6$  iterations of part 2 of the algorithm.

The next section gives a variant of the Adleman algorithm which improves the situation for  $GF(2^{127})$ .

### 3. The Waterloo Algorithm

This algorithm (see [3]) differs from the former algorithm in two ways. In part 2 where the polynomial  $h(x) = g(x)\alpha^a(x)$  is factored this algorithm applies the extended Euclidean algorithm to the polynomials  $h(x)$  and  $f(x)$  ( $f(x)$  is the irreducible polynomial of degree  $n$  which defines  $GF(2^n)$ .) so that we can write

$$h(x) = \frac{s(x)}{t(x)}$$

where  $(s(x), t(x)) = 1$  and  $\deg s(x), \deg t(x) \leq \frac{n}{2}$ . One observation we can make at this point is that every polynomial  $h(x)$  of degree at most  $n$  ( $n$  odd) can be written uniquely as a quotient of polynomials where each has degree at most  $\frac{n}{2}$  and which are relatively prime. If both  $s(x)$  and  $t(x)$  factor into  $S$  then  $\log g(x)$  is readily computed by table lookup.

The advantage to this algorithm over the previous one is that it is more probable for two polynomials of degrees at most  $\frac{n}{2}$  to factor into the database than it is for one polynomial of degree  $n$ . Let  $N(b, i, j)$  be the number of pairs of relatively prime polynomials  $(A(x), B(x))$  such that  $\deg A(x) = i, \deg B(x) = j$  and both are smooth with respect to  $b$ . (By smooth we mean that the polynomial factors into irreducibles all of whose degrees are at most  $b$ .) For each irreducible polynomial  $b(x)$  with degree  $k \leq b$  define the enumerator of  $b(x)$  by

$$(1+y^k+y^{2k}+\dots+z^k+z^{2k}+\dots).$$

Letting  $l_k$  be the number of irreducible polynomials of degree  $k$  we obtain the generating function for the  $N(b, i, j)$  as

$$\begin{aligned} F(y, z) &= \prod_{k=1}^b \left( \sum_{i=0}^{\infty} y^{ik} + \sum_{i=1}^{\infty} z^{ik} \right)^{l(k)} \\ &= \prod_k [(1-y^k z^k)/(1-y^k)(1-z^k)]^{l(k)} \\ &= E(y)E(z)/E(yz) \end{aligned}$$

where  $E(y)$  is the generating function for one smooth polynomial. The probability that an ordered pair of relatively prime polynomials  $(A(x), B(x))$  each of degree at most  $\frac{n}{2}$  are both smooth with respect to  $b$  is

$$p^*(n, b) = \left\{ \sum_{0 \leq i, j \leq \frac{n}{2}} N(b, i, j) \right\} / 2^n.$$

In the case of  $GF(2^{127})$  Coppersmith [6] has evaluated this expression for  $b=17$  and found  $p^*(127, 17) \cong \frac{1}{7000}$ . In order to simplify calculations we approximate  $p^*(n, b)$  by  $[p(n, b)]^2$ . For  $GF(2^{127})$ ,  $[p(127, 17)]^2 \cong \frac{1}{5000}$ .

In the following table we list these probabilities for  $b$  ranging between 1 and 30 and also we list the probabilities associated with the Adleman algorithm. The table also includes the expected number of iterations to produce enough equations to construct the database for each value of  $b$  in the given range.

We see from the table that the number of iterations for the database is minimized by  $b=20$ . For our implementation we selected  $b=17$  in order to keep the system of equations manageable.

The second difference in the Waterloo algorithm is in producing the equations for the database. We did not rely entirely on Part 2 of the algorithm. A number of equations were readily obtained by several techniques which make use of the fact that squaring is a linear operator in the field. Our principle technique here is referred to as the generation of systematic equations.

We briefly describe this technique with regard to  $GF(2^{127})$  generated by  $f(x)=x^{127}+x+1$ . Since  $f(x)$  divides  $x^{128}+x^2+x$ , it is easily shown that  $x^{2^i}$ ,  $0 \leq i \leq 126$ , can be written as  $e = \sum_{j=0}^6 \gamma_j x^{2^j}$  where  $\gamma_j \in \{0, 1\}$  and so the log of any element of the form  $e$  can be readily found. Similarly the log of any element of the form  $\gamma_{-1} + \sum_{j=0}^6 \gamma_j x^{2^j}$  where  $\gamma_j \in \{0, 1\}$ ,  $-1 \leq j \leq 6$  can be found. This gives 31 equations where the maximum degree is 16. Related to this idea is the observation that if  $u(x)$  is any irreducible polynomial of degree  $d$  then the degrees of all irreducible polynomials of  $w(u(x))$  are divisible by  $d$ . Using this method we obtained 142 linearly independent equations involving the 226 logarithms of irreducible polynomials of degree  $\leq 10$ .

Probability and expected number of runs for the new and Adleman algorithm,  $n=127$ .

Degree	Total no. of irred. polys	New algorithm		Adleman algorithm	
		Probability	Expected no. of runs	Probability	Expected no. of runs
1	1	1.27141469E-32	7.8652544E+31	0.47772090E-34	0.20932720E+35
2	2	1.61023467E-30	1.24205499E+30	0.10391370E-32	0.19246730E+34
3	4	1.42003032E-27	2.81684126E+27	0.10686600E-30	0.37430020E+32
4	7	1.15313844E-24	6.07038995E+24	0.16022050E-28	0.43689770E+30
5	13	3.46321796E-21	3.75373429E+21	0.12806100E-25	0.10151400E+28
6	22	2.43083741E-18	9.05037906E+18	0.60489980E-23	0.36369650E+25
7	40	1.75009226E-15	2.28559379E+16	0.58661830E-20	0.68187380E+22
8	70	2.99279106E-13	2.33895379E+14	0.20402690E-17	0.34309190E+20
9	126	2.39477444E-11	5.26145586E+12	0.40463370E-15	0.31139250E+18
10	225	7.73424039E-10	2.90914154E+11	0.31781350E-13	0.70796220E+16
11	411	1.42517804E-08	2.88385022E+10	0.13612270E-11	0.30193290E+15
12	746	1.48157461E-07	5.03518348E+09	0.29555160E-10	0.25240930E+14
13	1376	1.0660927E-06	1.29069451E+09	0.41129290E-09	0.33455440E+13
14	2537	5.46899677E-06	463887639	0.37461550E-08	0.67722710E+12
15	4719	2.19028276E-05	215451634	0.24991280E-07	0.18882570E+12
16	8799	7.12191038E-05	123548311	0.12723630E-06	0.69154690E+11
17	16509	1.96662481E-04	83945854.4	0.52428530E-06	0.31488570E+11
18	31041	4.71015488E-04	65902291.5	0.17992970E-05	0.17251720E+11
19	58635	1.00764675E-03	58190035.4	0.53284450E-05	0.11004140E+11
20	111012	1.96260912E-03	56563479.1	0.13895770E-04	0.79888990E+10
21	210870	3.54177251E-03	59537985.4	0.32587350E-04	0.64709140E+10
22	401427	5.96621694E-03	67283339.5	0.69740460E-04	0.57560100E+10
23	766149	9.45338803E-03	81044911.9	0.13806940E-03	0.55490060E+10
24	1465019	.0142135998	103071637	2.25535770E-03	0.57373450E+10
25	2807195	.0204564492	137227872	0.44523690E-03	0.63048900E+10
26	5387990	.0283794933	189855046	0.73751900E-03	0.73054240E+10
27	10358998	.0381757153	271350462	0.11680410E-02	0.88686300E+10
28	19945393	.0500248252	398709899	0.17773340E-02	0.11222780E+11
29	38458183	.0640949758	600018684	0.26095230E-02	0.14737250E+11
30	74248450	.0805164802	922152208	0.33185230E-01	—

Another example of making use of the linearity of squaring is a method referred to as weight manipulation [4] (for more details see [5]). Define  $p(x)=x^m+g(x)$  where  $g(x)$  has degree  $k < m$ . Define  $d$  to be the largest integer such that  $2^d m \leq n$ . Let  $s=n-2^d m$  and consider

$$\ell(x)=x^s p(x)^{2^d} \pmod{f(x)}.$$

It follows that

$$\deg \ell(x) = \max \{k2^d, \deg(f(x)+x^n)\}.$$

If  $p(x)$  and  $\ell(x)$  are smooth with respect to  $b$  then we obtain an equation. For example, in  $GF(2^{127})$  generated by  $f(x)=x^{127}+x+1$  take  $p(x)=1+x^7$  and  $d=7$  then  $s=15$ . This gives

$$\ell(x)=x^{15}(1+x^7)^{16}=1+x+x^{15}.$$

Clearly, both  $\ell(x)$  and  $p(x)$  are smooth with respect to  $b=17$ . A similar result can be developed for

$$(p(x))^{2^{d+1}}.$$

Coppersmith [6] has extended the idea behind these techniques and has obtained striking improvements in the index calculus methods. We should mention that asymptotically the Waterloo algorithm is of the same order ( $\exp(c_2(n \log n)^{1/4})$ ) as the Adleman algorithm but for fields  $GF(2^n)$  of practical interest it gives much better running times. The Coppersmith algorithm is described in the next section.

The database for  $GF(2^{127})$  was constructed at Denelcor using their HEP computer [12]. A HEP consists of from one to sixteen process execution modules and each is a pipelined processor with a depth of 8. Since the HEP can run a given program in parallel with itself, the index-calculus algorithms are ideally suited to this architecture. Several copies of the algorithm were run in parallel to produce linear equations which when added to those found systematically and by weight manipulation would yield 16,510 linearly independent equations in the 16,510 unknown logs. Sixteen copies of the algorithm in parallel appeared to be optimal. The database took about 7 hours to build and computing individual logs using Part 2 takes under 1 second on the HEP. Having the database our implementation on a VAX11/780 takes about five minutes per logarithm.

As mentioned in the previous paragraph, the index calculus algorithms are ideally suited to parallel processing. If we run  $n$  copies of the program in parallel then the logarithm is obtained from the copy which finishes first. We compute the expected number of iterations to find the logarithm of an element given that  $n$  copies of the algorithm are running simultaneously. Let  $p$  be the probability that an iteration will succeed in computing the logarithm and let  $q=1-p$ . Suppose  $i$  of the processes complete on the  $k^{\text{th}}$  iteration and the remaining  $n-i$  do not. The probability of this event is

$$(pq^{k-1})^i \left(1 - \sum_{s=1}^k pq^{s-1}\right)^{n-i}$$

and the probability that at least one of the processes finishes on the  $k^{\text{th}}$  iteration is

$$\sum_{i=1}^n \binom{n}{i} [pq^{k-1}]^i [1 - \sum_{s=1}^k pq^{s-1}]^{n-i}. \quad (*)$$

Since  $p \sum_{s=1}^k q^{s-1} = 1 - q^k$ , (\*) equals  $q^{(k-1)n} [1 - q^n]$ . To compute the expected number of iterations we evaluate

$$\sum_{k=1}^{\infty} k q^{(k-1)n} [1 - q^n] = (1 - q^n)^{-1}.$$

Since  $q=1-p$  then it follows that  $(1 - q^n)^{-1} \cong (np)^{-1}$  as one might expect. This expected number of iterations did occur when we made experimental runs on the HEP.

#### 4. The Coppersmith Algorithm

The algorithm described in this section makes extensive use of the linearity of squaring in  $GF(2^n)$ . The method applies also to  $p^h$  powers in  $GF(p^n)$  for  $n > 1$ .

Coppersmith's modification [6] of this basic index-calculus algorithm of section 2 improves substantially the performance of both parts 1 and 2. Let's first consider part 1.

The algorithms of the previous sections and the algorithm given here rely on the fact that polynomials of degree  $k$  tend to factor into polynomials whose degrees are "small". In order to generate enough equations to construct the database for  $GF(2^n)$  select pairs of polynomials  $(a(x), b(x))$  such that  $\deg a(x) \leq d$  and  $\deg b(x) \leq d$  where  $d < b$  and the gcd of  $a(x)$  and  $b(x)$  is 1. It is easily seen that there are precisely  $2^{2d+1}$  such pairs. Let  $k$  be a power of 2 near  $\sqrt{n/b}$  and choose  $h$  to be the least integer greater than  $\frac{n}{k}$ . Suppose also that the generating polynomial for the field has the form  $f(x) = x^n + g(x)$  where  $g(x)$  has low degree. Let  $h(x) = x^{hk} = g(x)x^{hk-n}$  and define

$$c(x) = x^h A(x) + B(x)$$

and

$$d(x) = [c(x)]^k.$$

If both  $c(x)$  and  $d(x)$  are smooth with respect to  $b$ , then we obtain an equation for the database. In the case  $n=127$  we take  $b=17$ ,  $k=4$ ,  $h=32$  and  $d=10$ . The polynomials  $c(x)$  and  $d(x)$  have degrees  $\leq 42$ . Coppersmith [6] shows that the 2 million possible pairs  $(a(x), b(x))$  yield about 47,000 equations in the 16,510 unknowns. The above procedure is a variant on the weight manipulation method of the previous section. A more direct application of weight manipulation can be described. We illustrate the technique in the case  $GF(2^{127})$  generated by  $f(x) = x^{127} + x + 1$ . Select pairs of polynomials  $(a(x), b(x))$  which are relatively prime and such that  $\deg a(x) \leq 7$  and  $\deg b(x) \leq 8$ . Form the polynomials  $c(x) = a(x)x^{31} + b(x)$  and  $d(x) = x^3[c(x)]^4$  where  $\deg c(x) \leq 38$  and  $\deg d(x) \leq 35$ . If  $c(x)$  and  $d(x)$  are smooth with respect to  $b=17$  we get an equation.

We now describe part 2 of the algorithm. Let  $g(x)$  be a field element whose log is to be found. Suppose  $t = \deg g(x)$ . Let  $k$  be a power of 2 close to  $\sqrt{n/t}$  and  $h$  be the least integer greater than  $n/k$ . For example, if  $n=127$ ,  $b=17$  and  $t=33$  then choose  $k=2$  so that  $h=64$ . Finally select  $d$  close to  $(t + \sqrt{n/b}) \log n / 2$ . In our example we take  $d=23$ . Choose a relatively prime pair of polynomials  $a(x)$  and  $b(x)$  each of degree  $\leq d$  such that  $g(x)$  divides  $c(x) = x^h a(x) + b(x)$ . The choices for  $a(x)$  and  $b(x)$  can easily be determined by solving a linear system of equations over  $GF(2)$ . Let  $d(x) = [c(x)]^k$ . If both  $c(x)$  and  $d(x)$  are smooth with respect to the bound  $b^{(1)} = h\sqrt{b/n}$  then we have at most  $2t$  irreducible factors of degree at most  $b^{(1)}$ . For each factor with degree  $> b$  we iterate this procedure and reduce the bound  $b^{(1)}$  with each iteration. That is, at iteration  $i$ ,  $b^{(i)} = h(b/n)^{-2^i}$ . If not both of  $c(x)$ ,  $d(x)$  are smooth then choose a new pair  $a(x)$ ,  $b(x)$  and start again.

The asymptotic running time of this algorithm is computed by Coppersmith [6] to be  $\exp(cn^{\frac{1}{3}}(\log n)^{\frac{2}{3}})$  which improves the results of the previous two sections.

We have implemented part 2 of Coppersmith's algorithm. The following table displays test results of 8 samples each consisting of 100 randomly generated polynomials of degree at most 126. The degree bound  $b$  is always 17 but for each sample we varied the Coppersmith degree bound  $b'$  and the value of  $d$ . We have a column labelled "Total Number of Basic Iterations". This refers to the number of iterations required to obtain smoothness with respect to  $b'$ . The column labelled "Total Number of Coppersmith Iteration" is the total number of iterations to obtain smoothness with respect to degree bound 17 for those polynomials with degree between 17 and  $b'$ . We also found that in this case a single step reduction of the Coppersmith reduction was optimal.

$b'$	$d'$	Total number of basic iterations	Total number of Coppersmith iterations	Average time in sec per log
20	12	99,496	6,731	57
20	13	72,519	7,262	44
21	12	98,245	9,087	49
21	13	39,327	9,995	19
22	13	38,390	11,859	28
22	14	24,741	16,299	28
23	14	20,296	20,726	27
23	15	12,857	27,045	47

From the table it appears that  $b'=23$  and  $d=14$  is optimal. With these values we ran a sample of 500 randomly generated polynomials with the following result. The column headings are the same as in the previous table.

23	14	120,661	101,563	26.6
----	----	---------	---------	------

For  $GF(2^{127})$  we refer to an equation of the form

$$[C(x)]^4 = [A(x)x^{32} + B(x)]^4$$

as a Coppersmith equation. As a direct generalization of the weight manipulation technique discussed earlier we consider equations of the form

$$x^i[C(x)]^4 = x^{127-4i}[A(x)x^i + B(x)]^4$$

and refer to these as *underflow* equations. With the degree of  $A(x)$  and  $B(x)$  at most 9 in the Coppersmith equations we generated 19461 equations for the database in 8.1 hours on the VAX. With the degrees of  $A(x)$  and  $B(x)$  at most 8 we obtained 7777 equations in 2.23 hours. The following table lists results when underflow equations were used. The columns labelled *deg A* and *deg B* refer to the highest degree used for  $A$  and  $B$  respectively. For polynomial  $A$  we require that it has a constant term so that no Coppersmith equations are generated this way and the equations obtained from distinct rows are all different.



<i>deg A</i>	<i>deg B</i>	<i>i</i>	Value of Equations	Number of (Hours)
8	9	31	7891	2.89
9	8	30	7193	2.88
10	7	29	5975	2.80

If we use the Coppersmith equations with the degrees of  $A(x)$  and  $B(x)$  at most 8 and we use the underflow equations given in the first two rows of the table we get 22,861 equations in 8 hours or one equation every  $.3499 \times 10^{-3}$  hours. This is approximately a 16% improvement over collecting the 19461 equations using the Coppersmith equations with  $\deg A(x)$ , and  $\deg B(x) \leq 9$ . This data was obtained by running our program on a VAX11/780 at the University of Waterloo. The program is written in Fortran 77 with the gcd routines in assembler.

## 5. Conclusion

We have described the basic index-calculus algorithm and two variants of it. The basic algorithm and the Waterloo algorithm both carry over to  $GF(p)$ ,  $p$  a prime. Using the Euclidean algorithm for integers it is easy to show that given an integer  $a$ ,  $1 \leq a \leq p-1$ , that there exist integers  $s$  and  $t$  such that  $as \equiv t \pmod{p}$  and  $1 \leq |s|$ ,  $t \leq p-1$ .

As for  $GF(2^n)$  it is clear that for  $n=127$  this field is insecure and should be avoided in cryptographic schemes. In [13] Odlyzko analyses the performance of the Coppersmith algorithm for various values of  $n$  running on various types of equipment. The conclusion seems to be that an ambitious effort might be able to produce the necessary database for  $n \leq 1280$ .

## References

- [1] L.M. Adleman, A subexponential algorithm for the discrete logarithm problem with applications to cryptography, *Proc. 20th IEEE Found. Comp. Sci. Symp.* (1979), 55-60.
- [2] B. Arazi, Sequences constructed by operations modulo  $2^n-1$  or mod  $2^n$  and their application in evaluating the complexity of a log operation over  $GF(2^n)$ , preprint.
- [3] I.F. Blake, R. Fujii-Hara, R.C. Mullin and S.A. Vanstone, Computing logarithms in finite fields of characteristic two, *SIAM J. Alg. Disc. Methods*, Vol. 5 #2 (1984), 276-285.
- [4] I.F. Blake, R. Fujii-Hara, R.C. Mullin and S.A. Vanstone, Finite field-techniques for shift registers with applications to ranging problems and cryptography, Final Report Project #106-16-02, Department of Communications (1983).
- [5] I.F. Blake, R. Fujii-Hara, R.C. Mullin and S.A. Vanstone, An attack on the discrete logarithm problem in  $GF(2^{127})$ , Progress Report, Project #106-16-02, Department of Communications (1982).
- [6] D. Coppersmith, Fast evaluation of logarithms in fields of characteristic two, *IEEE Trans. Inform. Theory*, (July 1984), 587-594.
- [7] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, IT-22 (1976), 644-654.

- [8] T. ElGamel, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, to appear.
- [9] T. Herlestam and R. Johanneson, On computing logarithms over  $GF(2^p)$ , *BIT* 21 (1981), 326-334.
- [10] D.E. Knuth *The Art of Computer Programming: Vol. 2. Seminumerical Algorithms*, 2nd ed. Addison-Wesley 1981.
- [11] D.L. Long and A. Wigderson, How discreet is the discrete log? *Proc. 15th ACM Symp. Theory of Computing* (1983), 413-420.
- [12] R.C. Mullin, E. Nemeth and N. Weidenhofer, Will public key crypto systems live up to their expectations? HEP implementation of the discrete log codebreaker, preprint.
- [13] A. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, *Eurocrypt-84* (to appear).
- [14] S.C. Pohlig and M. Hellman, An improved algorithm for computing logarithms over  $GF(p)$  and its cryptographic significance, *IEEE Trans. Inform. Theory* IT-24 (1978), 106-110.
- [15] B.P. Schanning, Data encryption with public key distribution, *EASCON Conf. Rec.*, Washington D.C., October 1979, 653-660.
- [16] A.E. Western and J.C.P. Miller, Tables of indices and primitive roots, *Royal Society Mathematical Tables*, Cambridge University 9 (1968).
- [17] K. Yiu and K. Peterson, A single-chip VLSI implementation of the discrete exponential public key distribution system, *Proc. GLOBECOM-82, IEEE* (1982), 173-179.

WYNER'S ANALOG ENCRYPTION SCHEME:  
RESULTS OF A SIMULATION

Burt S. Kaliski

MIT Laboratory for Computer Science  
545 Technology Square  
Cambridge, Massachusetts 02139

ABSTRACT

This paper presents the results of a simulation of an analog encryption scheme. The scheme, introduced in 1979 by Aaron Wyner of Bell Telephone Laboratories, provides secure, accurate scrambling of speech waveforms, while conforming to the bandlimitedness of a telephone channel. The simulation confirms the scheme's theoretical properties, based on numerical measures and on listening to encrypted and decrypted waveforms.

INTRODUCTION

Security in communications is increasingly important. While the thrust of research is in digital encryption methods, analog encryption is also of growing interest. Such encryption is useful, for example, in transmission to mobile telephones in automobiles, and to cellular radios.

Analog encryption for telephone speech poses special problems. Unlike digital encryption, it must be performed in real time. The result of encryption must conform to the bandwidth of the telephone channel, so that no information is lost. And it must be secure.

A scrambling scheme introduced by Aaron Wyner of Bell Telephone

Laboratories [Wyner, 1979a and 1979b] answers a theoretical question: Are bandlimitedness and security mutually exclusive? The answer is a theoretical "yes"; we present here a practical "yes," as confirmed by a software simulation of the scheme.

Let us now define a scrambler, since it is a term we will use frequently. One may view a scrambler as a black box whose input is an element of one space, and whose output is an element of another. The black box computes a one-to-one mapping from the input space to the output space, based on some secret quantity, or key.

Associated with the scrambler is a descrambler, a black box that computes a one-to-one mapping from the output space to the input space. The mapping is the inverse of that of the scrambler, when the secret quantity or key is the same. Without the key, however, it is computationally difficult to compute the inverse.

The input and output spaces may be bit-strings of a given length, or a given number of samples of an analog signal. Although the samples of an analog signal can be considered as bit-strings (for instance, by concatenating the bits representing the samples), we choose to differentiate between analog and digital scramblers, based on the interpretation of the inputs and outputs.

An practical scrambler, digital or analog, must have two characteristics: security and speed. An analog scrambler must also be accurate. With regard to the model here, such a scrambler must conform to the bandlimitedness and noise properties of a telephone channel. Speed also is critical, because an analog scrambler for telephone communication must operate in real time.

## RESULTS OF THE SIMULATION

The simulation of Wyner's speech scrambler gives good results. Accurate decryption and secure encryption are shown to be achievable with reasonable processor power. In particular, a signal-to-noise ratio of about 13 dB is possible with a processor that can perform a multiply-and-add in 4  $\mu$ s.

Several parameters define the operation of the scrambler. These parameters are adjusted to conform to the characteristics of the telephone channel, the quality of output desired, and the available processor power for the scrambler. A set of parameters that provides high-quality output and requires reasonable encryption speed for a

typical telephone channel is the following:

- o block size -- 32 points
- o sampling rate -- 8000 per second
- o input band -- [0, 2700]
- o output band -- [300, 3200]

The significance of each of these parameters is described below.

The main test of the scrambler is the encryption and decryption of an utterance of the word "potato." A qualitative, and somewhat subjective, analysis of the scrambler provides encouraging results.

The decrypted signal sounds clear and accurate, with little difference in quality from the original signals. The transmitted signal, as expected, resembles white noise. One is able, however, to identify vowels when hearing the waveform. The signal sounds like "zhuh-zhuh-zhuh," one "zhuh" for each vowel. This problem, a result of orthogonality in the scrambler, is discussed below.

Figures 1 and 2 show the waveforms and spectrograms for the transmitted and decrypted signals. The figures are hardcopies of a Lisp machine display using a speech analysis system called Spire, which was developed at the MIT Speech Laboratory.

The transmitted signal's spectrogram is distributed nearly uniformly across the desired band, but not across time. Note that its short-time energy resembles that of the decrypted signal. The decrypted signal's spectrogram looks very much like speech, with the vowels (O and A) clearly visible.

The results, although favorable, are not complete. Certain simplifications in the software model lead to results perhaps better than those attainable in practice. Nonetheless, high quality with reasonable processor power is still possible. The simplifications are explained below.

#### CHOICE OF PARAMETERS

Four parameters determine the operation of the scrambler. Their values determine the running time of the scrambling algorithm, the accuracy of transmission, and the level of security achieved.

<untranscribed>

<antrænskra<sup>y</sup>bd>

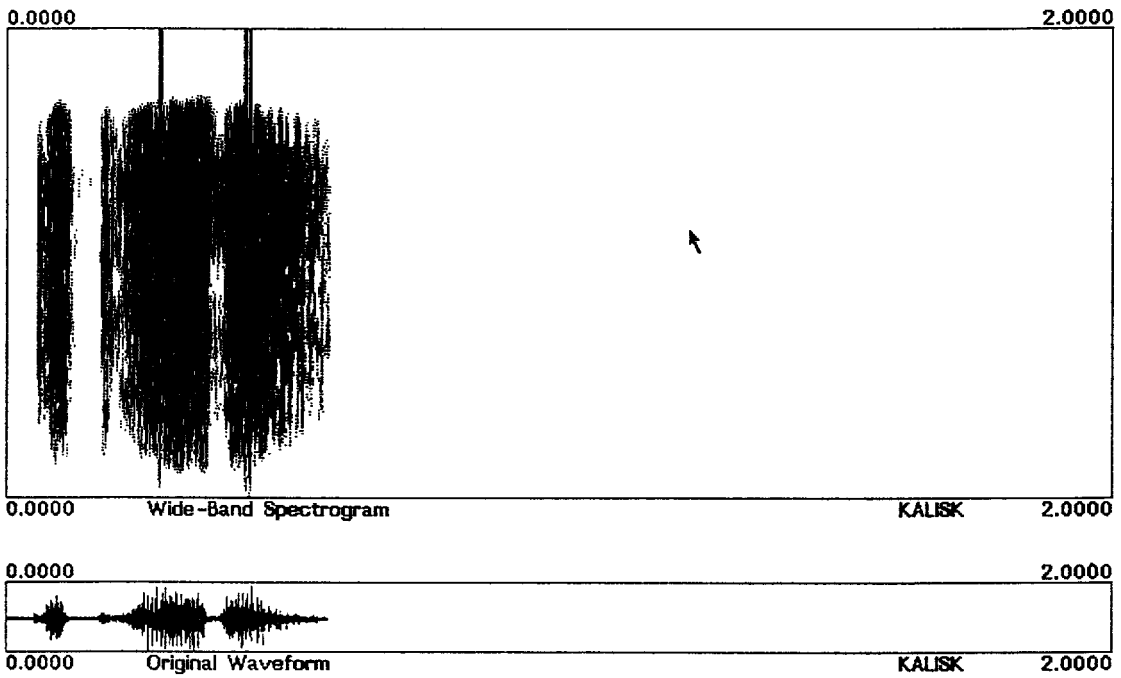
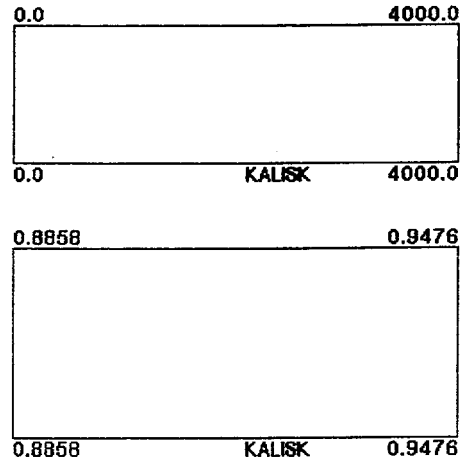


Figure 1 -- Spectrogram of trasmitted waveform

<untranscribed>

<untranscribed>

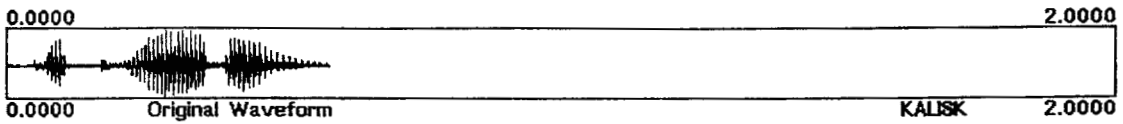
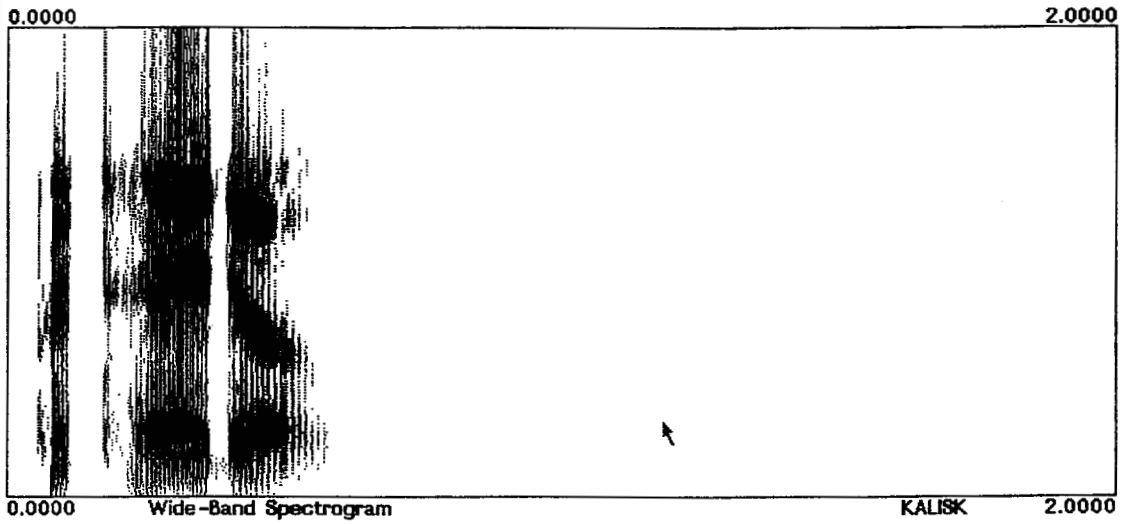
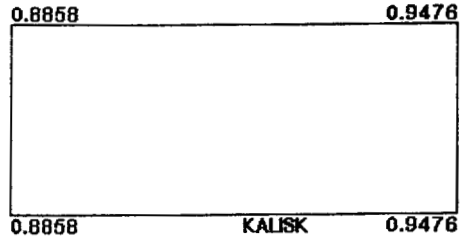
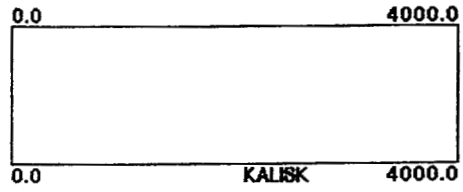


Figure 2 -- Spectrogram of decrypted waveform

- o block size -- The scrambler operates on blocks of samples, producing N outputs for every N inputs. The block size determines the security, and also the speed requirements. Typical values might be 32, 64, or 96. We choose 32 to minimize speed requirements.
- o sampling rate -- A new sample of the input waveform is taken every T seconds. For speech, a rate of 8 kHz ( $T = 125 \text{ us}$ ) is about the minimum to avoid aliasing. Higher rates are generally not required, but they may improve the quality of the decrypted signals.
- o input band -- This is the frequency band used to determine the basis vectors for scrambler input (and hence descrambler output). It is generally selected to maximize the amount of input energy included. For small N, much energy appears as DC, so a low end of 0 is chosen. The high end is at 2700 to provide a bandwidth narrower than that of the output.
- o output band -- This is the band used to determine basis vectors for scrambler output and descrambler input. It is generally wider than the input band, and conforms to the characteristics of the telephone channel. We use [300, 3200], corresponding to the simulator's channel model.

Design of most of the scrambler depends not on the actual frequencies--such as [300, 3000]--but on their discrete equivalents, those scaled by the sampling period. With the parameters above, the equivalent band would be [.0375, .375]. In the discussion that follows, both input and output bands are loosely referred to as [W1, W2], where W1 and W2 are in the range [0, .5]. The quantity W represents the bandwidth, or  $W2 - W1$ .

## HOW THE SCRAMBLER WORKS

The bandlimitedness of the telephone channel constrains the output space of the black box scrambler model. Only those outputs resembling speech in bandwidth may be produced. Similarly, only those inputs of such form should be expected. Thus a mapping between the speech-like subspaces of the sets of all samples must be provided.

We can approximate those subspaces by saying they correspond to the band [W1, W2] within the frequency spectrum on [0, .5]. Thus, while the spaces obtained from N independent samples may have



dimension  $N$ , the subspaces--constrained to be near zero outside the selected band--really have dimension  $2WN$ . With the parameters above, this quantity is about 22 for the input band, and 23 for the output band.

The definition of bandlimited also includes indexlimited in the context of a speech scrambler. Each block of samples, if considered alone with samples in all other blocks set to zero, should be bandlimited for sufficiently large  $N$ . This means the discrete-time Fourier transform (that is, the transform from an infinite set of discrete values to an infinite, continuous waveform) of the block considered alone, should be roughly limited to  $[W1, W2]$ .

Assuming that the subspace we want is of dimension  $2WN$ , we must find a way to describe it. A naive description involves the discrete Fourier transform (that is, the transform from a set of discrete values to a set of discrete values). Perhaps the subspace is all sets of samples whose DFT is zero, except at those  $2WN$  points in the desired band. Since the transform is one-to-one, the subspace would be the correct size.

The solution is not quite so simple. We seek sequences that are both indexlimited and bandlimited. Those whose DFT is zero, except at certain points, include sine waves. Certainly these are not indexlimited.

Mathematical physics has a set of functions called prolate spheroidal waveforms. These waveforms are the only eigenfunctions of the finite Fourier transform (that is, the transform from a continuous, infinite waveform to a continuous waveform over  $[0, 1]$ ). Recall that eigenfunctions are those that, when transformed, are but scaled over a certain range--in this case,  $[0, 1]$ .

The discrete counterparts of the waveforms--discrete prolate spheroidal sequences--are similarly related to the discrete-time Fourier transform. Here the relationship is somewhat different. The sequences are "eigensequences" not of the transform itself, but of bandlimiting and of indexlimiting.

Specifically, when a sequence, defined by parameters  $N, W1$  and  $W2$ , is bandlimited to  $[W1, W2]$ , it is only scaled in indices  $[1, N]$ . The scaling is by the eigenvalue corresponding to the eigenvector.

Recalling digital signal processing, it is seen that to bandlimit a sequence is the same as to convolve the sequence with the non-causal impulse response of an ideal filter. For applying the response to a finite set of points, this is equivalent to

multiplication by a matrix derived from the impulse response. The eigenvectors of such a matrix are the eigensequences above.

The key result is that the eigenvalues fall into three categories: those close to 1, those close to 0, and others between 1 and 0. The eigenvalue represents the amount energy in the desired band  $[W1, W2]$ . The majority of values are close to 1 or 0;  $2WN$  such values are close to 1. The corresponding  $2WN$  eigenvectors are the basis of the subspace.

An example of prolate spheroidal sequences is given in Figure 3. The sequences, their discrete-time Fourier transforms and their eigenvalues are shown, with  $N = 8$  and band  $[.0375, .375]$ . Notice the high concentration of the first four sequences, and the low concentration of the last two.

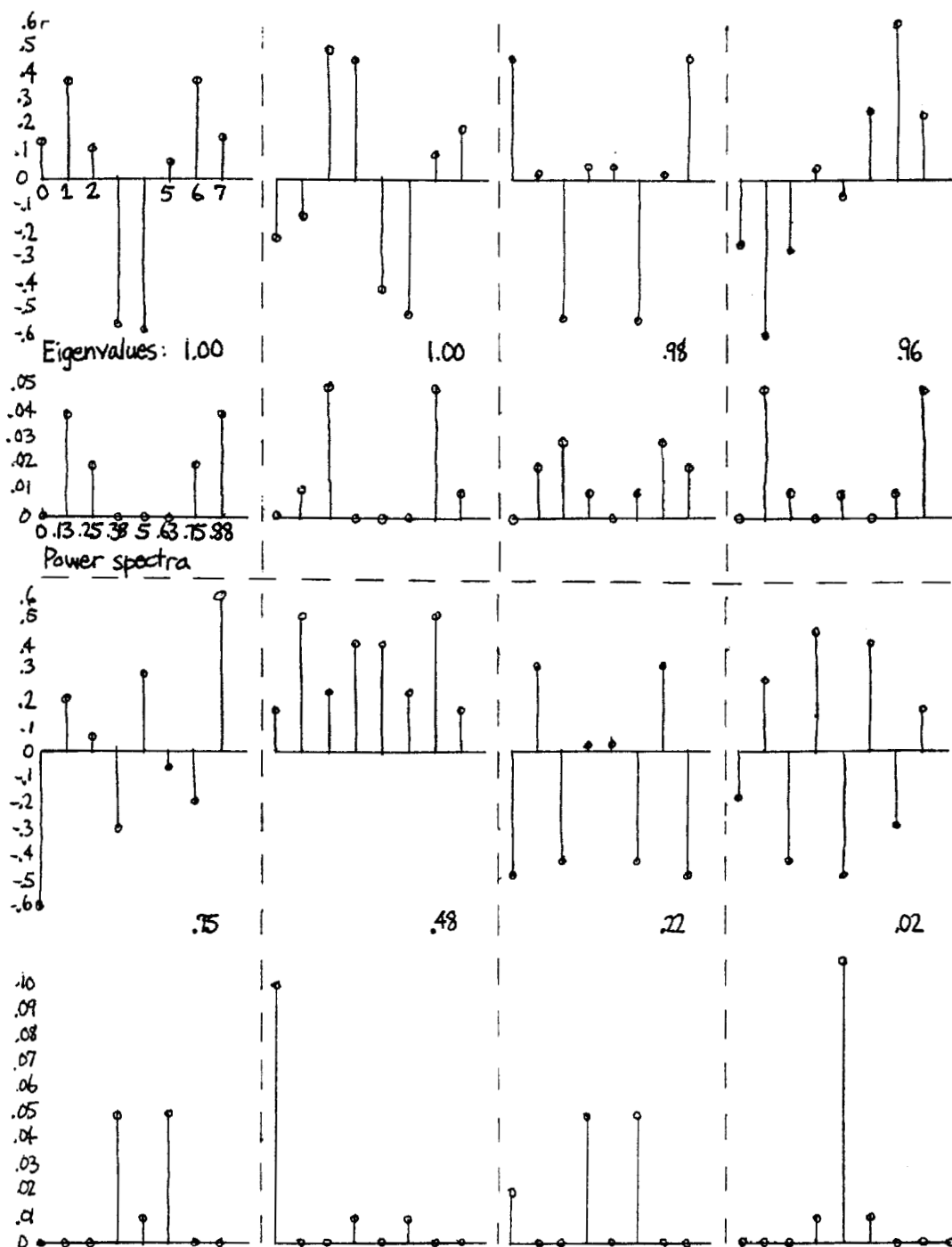
It remains to be shown how to use these discrete prolate spheroidal sequences to scramble speech. Since  $2WN$  sequences form an approximate basis for the desired subspace, we can compute for some set of  $N$  samples a weight corresponding to each sequence. The weights are easily found by dot-products.

We know that any combination of the  $2WN$  sequences will remain bandlimited. Hence the weights computed may be scrambled in any fashion, and a new set of samples produced. Scrambling the weights by an orthogonal transformation is most accurate, because any error in computing the weights is not increased.

The security comes from the orthogonal transformation. Multiplying the weights, represented as a  $2WN$ -element vector, by a  $2WN \times 2WN$  matrix, is equivalent. The matrix may be constructed randomly, based on a sequence of random numbers initiated by a secret key. The receiver, knowing the key, generates the same sequence of random matrices, and hence can reverse the orthogonal transformations.

One unfortunate result of the orthogonality is that the short-time energy of the output of the scrambler is the same as that of the input. This allows an adversary to differentiate between vowels and consonants. This problem is easily solved by adding a dummy waveform to the output to maintain a constant energy.

Considering the scrambling scheme as a series of matrix multiplications, we can estimate a running time. The transformation to prolate spheroidal weights is a  $2WN \times N$  matrix by  $N \times 1$  vector product. The  $2WN \times 2WN$  matrix by  $2WN \times 1$  vector product follows, then reconstruction using a  $N \times 2WN$  matrix by  $2WN \times 1$  vector product.

Figure 3 -- Prolate spheroidal sequences for  $N = 8$

The series of multiplications is equivalent to an  $N \times N$  matrix by  $N \times 1$  vector product. This requires  $N^2$  multiply-and-add operations for every  $N$  samples. Hence one such operation is required each  $T / N$  seconds, giving the 4 us timing above.

The simulator, of course, is not so constrained. It operates about 4000 times more slowly than a real-time scrambler. To scramble "potato" took several hours.

#### SIMULATING A MODEL OF THE SCRAMBLER

Notice the description of the scrambler above said very little about its role in a complete transmission system. Here we describe the way in which the black box is connected to the world--the input, transmission, and output processing.

Figure 4 contains a block diagram of a model given by Wyner and used in the simulation. The diagram has eight components. The scrambler and unscrambler are those described above. The channel is a typical telephone transmission channel; the equalizer is typical tapped delay-line used to compensate for linear distortion in the channel.

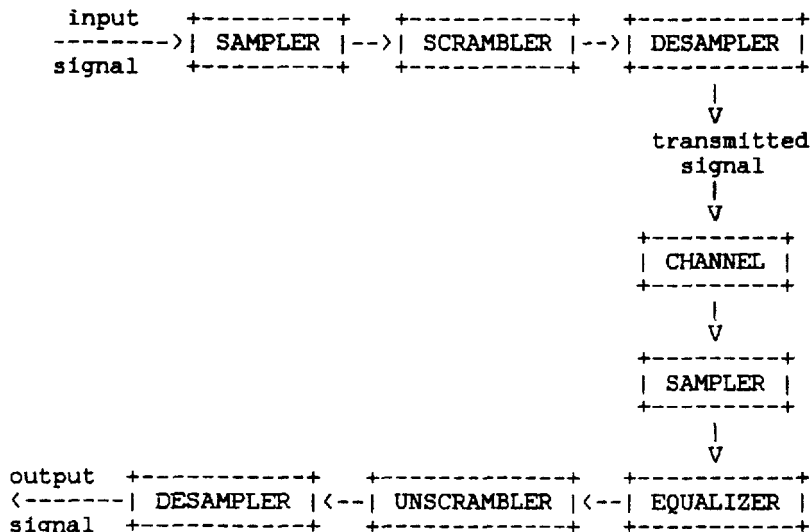


Figure 4 -- Block diagram of model for scrambler

The model of the scrambler naturally lends itself to object-oriented programming. Each module may be represented by a unique data type; indeed, even the signals and prolate spheroidal eigenvectors may be data types. Such an implementation is easily accomplished.

A Lisp Machine--a minicomputer designed specifically for programming in Lisp--is used for the simulation. Its system of "flavors" allows construction of the data types to make a block diagram, and its graphics capabilities are useful for viewing waveforms and sequences as they pass through the block diagram.

While the Lisp Machine is a sequential processor, the "transmission" of signals between modules is like that of a parallel, multiprocessor data-flow network. A driver passes input signals to the first module, which transmits to other modules, and so on, until the signals propagate to modules with no successors.

It should be noted that the test environment is incomplete. The simulated equalizer is built knowing the response of the telephone channel. In practice, the equalizer would not be able to compensate as well for linear distortion, and errors would be larger. In addition, most calculations in the simulator are performed with high-precision floating point numbers. In practice, fewer bits may be used, and quantization errors in sampling will be present. These factors are not included in the simulation.

## WHERE CRYPTOGRAPHY FITS IN

It may appear that the scrambler is based more on signal processing than on cryptography. The selection of scrambling matrices, however, is a problem of current interest in cryptography. A good overview of present methods for creating scrambling matrices is found in [Sloane, 1983].

Two general methods for selecting the random matrices come to mind. Precomputation with random selection from a set of such matrices is fast; computation at run-time is secure. In either case, some sequence of random numbers is needed. The "key" to the black box determines this sequence.

The simulator uses precomputation, since the matrix selected has little effect on the accuracy of the system. A set of matrices is computed by the Gram-Schmidt algorithm--by orthogonalizing matrices containing normally-distributed random elements.

Hadamard matrices (that is, those of size  $m \times m$ , orthogonal, with all elements either  $1/m$  or  $-1/m$ ), are best suited for run-time calculation. Random permutations and sign changes, all quickly done, further increase the security.

## CONCLUSION

A faster implementation of the scrambler is necessary for further study, given the slowness of the software model. The slowness is due primarily to the object-oriented implementation, which is ideal for detailed, step-by-step testing, but impractical for system testing. Since construction of a hardware unit in a short time period would be difficult, use of a processor with a floating-point accelerator is a logical next step.

## ACKNOWLEDGMENTS

The author is grateful to MIT Professor Ronald L. Rivest for advising the undergraduate thesis that led to this paper, and to Aaron Wyner for helpful comments on the thesis. I also wish to thank my fiancée, Cathy Oehl, for her love and support. Most importantly, for the blessing of an MIT education and for the opportunities it has made possible, I thank the Lord Jesus, to Whom my career and life are dedicated.

## REFERENCES

- Wyner, A.D. (1979a) "An analog encryption scheme which does not expand bandwidth -- Part I: Discrete time," IEEE Trans. Inform. Theory, 25.
- Wyner, A.D. (1979b) "An analog encryption scheme which does not expand bandwidth -- Part II: Continuous time," *ibid.*
- Sloane, N.J.A. (1983) "Encrypting by random rotations," in "Cryptography," Lecture Notes in Computer Science, 149, Springer-Verlag, Berlin.

# ON ROTATION GROUP AND ENCRYPTION OF ANALOG SIGNALS

Su-shing Chen

University of Florida  
Gainesville, Florida 32611, U.S.A.

## INTRODUCTION

The problem of generating random elements in groups has direct application to cryptography. For instance, we like to know whether the DES permutations are random permutations of the  $2^{64}$  possible 64-bit words. The whole symmetric group is known to be generated by certain k-functions (7). Another example is the Wyner voice encryption scheme (12) which requires the production of large numbers of random real orthogonal matrices. N. J. A. Sloane has given a survey on this problem (11) which has led to the following questions for a given group  $G$ :

1. How does one generate elements of  $G$  at random ?
2. How can one test if certain given elements of  $G$  really are random ?
3. Does a given subset  $H$  generate the whole group  $G$  ?
4. If so, how long does it take ?

In this paper, we consider these questions for the orthogonal group  $O(d)$  for any positive integer  $d$ . By looking at the Lie algebra  $\mathfrak{o}(d)$  of  $O(d)$  and one-parameter subgroups of  $O(d)$ , we can find the generation of an arbitrary element in terms of one-parameter rotation groups in uniform fashion. The length of generation can be determined. Random elements are generated using random number generator on the real parameter space of each one-parameter subgroup.

The structural theory of Lie groups and other groups seems to be useful to cryptography. For groups which are not Lie, one may try to embed them into Lie groups. The transformation group theory and ergodic theory emerge to be also very useful. Ergodic theory can be considered to be a generalization of

the existing probabilistic and statistical methods. We certainly owe the idea to Shannon in his classic paper in 1949 (11) on the mixing property of two non-commuting operations on some space. We only have to find an operation which is the iteration of these two operations to a certain high degree to achieve relatively mixing situation in the space. We may point out that a quite simple group, such as the real numbers, can act on a space in a very complicated way to yield a good cryptosystem. RSA system can be considered in this manner (4) as a transformation semigroup.

#### ENCRYPTION OF ANALOG SIGNALS

Wyner's voice encryption scheme offers high fidelity and high security to encrypting voice signals over telephone lines. The technique is applicable to other analog signals. For the space of approximately bandlimited sequences  $(a(1), \dots, a(N))$ , there is known basis  $x_1, \dots, x_d$ ,  $d \approx 2WN$ , where  $W < 1/2$  is the bandwidth, called discrete prolate spheroidal sequences (11). Each waveform is sampled every  $T$  seconds, where  $T$  is less than the Nyquist rate. We take a finite segment  $a = (a(1), \dots, a(N))$  of the sampled sequence and express it by

$$a = \sum_{j=1}^d a_j x_j,$$

where the coefficients are determined in the standard way and  $N$  is large enough to contain most of the energy in the given waveform.

The scrambling or encrypting is performed by multiplying the coefficient vector  $(a_1, \dots, a_d)$  by a secret  $d$  by  $d$  orthogonal matrix  $Q$ , obtaining

$$(b_1, \dots, b_d) = (a_1, \dots, a_d)Q.$$

The encrypted sequence is

$$b = \sum_{j=1}^d b_j x_j,$$

from which the encrypted waveform can be formed. The encrypted waveform has the



same bandwidth and approximately the same energy as the original waveform. Wyner has shown that if  $N$  and  $d$  are large enough and matrices  $Q$  are chosen independently and uniformly from the orthogonal group, then his scheme offers essentially perfect security (11), (12).

## THE ORTHOGONAL GROUP

The orthogonal group  $O(d)$  acting on the  $d$ -dimensional Euclidean vector space can be characterized by the preservation of the Euclidean inner product. The group  $O(d)$  is a Lie group and a Lie subgroup of the general linear group  $GL(d)$  of all nonsingular matrices. The manifold (Lie group) structure of  $GL(d)$  comes from the Euclidean vector space  $R^{2d}$  as an open subset. For each Lie group  $G$ , there is a Lie algebra  $\mathfrak{g}$  which is a vector space together with a Lie product  $[ , ]$ . For matrix Lie groups, the Lie product is the commutator  $[X, Y] = XY - YX$ , where  $X$  and  $Y$  are just matrices in  $R^{2d}$ . It is easy to see that the Lie algebra of  $GL(d)$  is  $R^{2d}$ . The key point is that the exponential mapping  $\exp(tX)$  brings an element  $X$  in the Lie algebra  $\mathfrak{g}$  to an element  $\exp(tX)$  in the Lie group  $G$ . For matrix Lie groups,

$$\exp(tX) = 1 + (tX) + (t^2X^2)/2! + \dots,$$

where  $X$  is any element of in  $\mathfrak{g}$ . The set of  $\exp(tX)$  for all  $t$  in  $R$  is a one-parameter subgroup in  $G$  and the derivative of  $\exp(tX)$  at  $t=0$  is  $X$ . Thus  $X$  is the velocity at  $t=0$  of the group  $\exp(tX)$ .

The orthogonal group  $O(d)$  and its Lie algebra  $\mathfrak{o}(d)$  are well known including their complete structures and associated mathematical invariants. The group  $O(d)$  is not connected and has two connected components; those having determinant  $+1$  form  $SO(d)$ , while those having determinant  $-1$  form the other component which is not a group because the identity matrix is not in it. The dimension of  $SO(d)$  as well as  $O(d)$  is  $d(d-1)/2$ . The dimension of the Lie algebra  $\mathfrak{o}(d)$  is that of  $O(d)$ . Indeed, the Lie algebra  $\mathfrak{o}(d)$  is also the Lie algebra of  $SO(d)$ . The classification of Lie subgroups of a simple Lie group is important to our consideration (3).

## RANDOM GENERATION

If we take a set of one-parameter groups  $\exp(tX_1), \dots, \exp(tX_m)$ , all the finite products of elements from these groups generate an arcwise connected subgroup of  $SO(d)$ . By Yamabe's theorem, it is a Lie subgroup. For  $m$  large enough, it contains a maximal subgroup and has to be the whole group  $SO(d)$ . For example,  $SO(3)$  has no 2-dimensional Lie subgroup and  $m = 2$  will be sufficient. Thus up to one-parameter subgroups, question 3. of Sloane is answered. Question 4. is answered by finding a positive integer  $n$  such that every element of  $SO(d)$  can be expressed as a product of elements selected from the given set of one-parameter subgroups and the length is at most  $n$  and by determining the minimal  $n$  over the collection of all such sets of one-parameter groups. Note that  $n$  depends on the choice of the set of one-parameter groups.

The first part is answered easily. Let  $S_n$  be all products of lengths less than or equal to  $n$ .  $S_n$  is compact and  $SO(d)$  is the union of  $S_n$  for all positive  $n$ . By the Baire category theorem, some  $S_m$  contains an open set whose translates cover  $SO(d)$ . This open cover has a finite subcover and the result is proved. To determine the number  $n$  for a given set of groups is rather complicated. One needs to study the geometry of the  $(d-1)$ -sphere  $S^{d-1}$  on which  $SO(d)$  acts and uses some mathematics in (3). The minimal number  $n$  can be shown to be  $d(d-1)/2$ .

In order to generate random elements, we generate random numbers on the real parameters. First, we generate random numbers on the interval  $[0,1]$ . By iteration, we get the whole real parameter space. The transition from  $SO(d)$  to  $O(d)$  is easy. The index of  $O(d)$  over  $SO(d)$  is two.

Since a direct product of low dimensional orthogonal groups is a subgroup of a higher dimensional one, we may increase the speed of encryption by segmenting the signal, applying lower dimensional groups and globally scrambling segments. This scheme provides a trap door for the system.

The advantage of our encryption system is that the set of one-parameter groups is fixed once for all as well as the form of the random orthogonal matrix.

This constancy is useful to hardwire the box to eliminate the intensive computation for generating a random orthogonal matrix. The pseudo-random scheme (11) using Hadamard matrices does not seem to generate truly random matrices. It is proved that for  $d$  greater than or equal to 8, the subgroup  $\bar{G}$  generated by the set  $S$  is topologically dense in  $O(d)$ , where  $S$  is a certain set of matrices (11). For  $d = 8$ , the cardinality of  $S$  is 4954521600. For  $d = 48$ , the cardinality becomes  $1.1765 \dots$  times  $10^{146}$ . We like to point out that the length of a finite product in the topological closure may have to be extremely large.

Finally, we may introduce other trap doors to our system to increase the speed of encryption, but our opponent still need to decrypt the signal randomly. One may use pseudo-random number generators, partial products of one-parameter subgroups or other structural constraints of Lie groups. The structure of Lie groups is very elegant and simple on one hand as we have seen above. On the other hand, extremely hairy situation may occur. For instance, we can embed a free group of two generators in a compact Lie group. Then this subset contains a free subgroup of infinitely many generators. Anyway, I agree with G. R. Blakley (2) that group theory offers a lot of opportunity for cryptologists to explore.

## REFERENCES

1. R. Bernhard, Breaching system security, IEEE Spectrum, 19 (1982), 24-31.
2. G. R. Blakley, Information theory without the finiteness assumption, I: Cryptosystems as group-theoretic objects, Crypto '84 Conference.
3. S. Chen and L. Greenberg, Hyperbolic spaces, Contribution to Analysis, Papers dedicated to L. Bers, Academic Press, 1974, 49-88.
4. S. Chen, Transformation groups and semigroups as cryptosystems, to appear.
5. S. Chen and R. Yoh, The category of generalized Lie groups, Trans. Amer. Math. Soc., 199 (1974), 281-294.
6. B. S. Kaliski, jr., Wyner's speech scrambler, Crypto '84 Conference.

7. A. G. Konheim, *Cryptography: A Primer*, John-Wiley & Sons, 1981.
8. N. R. F. MacKinnon, The development of speech encipherment, *Radio and Electronic Engineer*, 50 (1980), 147-155.
9. J. Reeds, Cracking a random number generator, *Cryptologia*, 1 (1977), 20-26.
10. D. Slepian, Prolate spheroidal wave functions, fourier analysis, and uncertainty, Part V: the discrete case, *Bell System Tech. Journal*, 57 (1978), 1371-1430.
11. N. J. A. Sloane, *Encrypting by random rotations*, Cryptography, T. Beth Ed., Springer-Verlag, *Lecture Notes in Computer Sciences*, 1983, 71-128.
12. A. D. Wyner, An analog scrambling scheme which does not expand bandwidth, Part I: discrete time, *IEEE Trans. Information Theory*, IT-25, No. 3, 1979, 261-274.

# THE HISTORY OF BOOK CIPHERS

Albert C. Leighton<sup>1</sup> and Stephen M. Matyas<sup>2</sup>

<sup>1</sup>Department of History  
State University of New York  
Oswego, New York 13126, U.S.A.

<sup>2</sup>Cryptography Competency Center  
IBM Corporation 69K/988  
Neighborhood Road  
Kingston, New York 12401, U.S.A.

## INTRODUCTION

You can do a lot with a book, besides read it! In fact, we know that by 1526--some 70 years after Gutenberg printed his first Bible--at least one of our forebears, Jacobus Silvestri, was thinking of how a book might be used for cryptographic purposes. Silvestri wrote of a sort of code book, or dictionary, which he recommended as a means to encipher written communications. From Silvestri, we can trace the development of book ciphers over a period of at least 400 years.

Book ciphers can be defined as any means of concealment in which a book, or existing text, is used as a basis for a cipher, from the simple and ridiculous to the complex and secure. They may include such oddities as Centos, in which words taken from the text are rearranged to form a different story; acrostics, which are built into a text and are extracted according to various rules to form a hidden meaning, and anagrams, in which the letters of a text are rearranged to form another text. The chief object of this paper is to demonstrate some ways in which book ciphers have been used for secure communications.

The simplest, and least secure method, is to take a dictionary and use the numbers of the page, column and line on which the desired word is found. This is equivalent to a one-part code in which both the numbers and the words run in their usual order. With such a simple method the approximate length of the dictionary and the location of particular initial letters in the dictionary can soon be ascertained.

Words may be cut from a text such as a newspaper or magazine and rearranged in Cento fashion to form a desired message. Handwriting and

other clues as to the origin of the message are concealed.<sup>1</sup> One of the oldest methods, used even by the ancient Greeks, makes use of tiny dots or pinpricks placed under the words or letters of a book to spell out a desired message.<sup>2</sup> The text is then sent to the recipient. The most effective of the book ciphers is the running key cipher where the cipher key is formed by the text of the book itself, which never repeats. As long as the key is used only once, the system is quite effective and reaches perfect security when the text used is random.

Centos are not practical for secret communication but are more likely to represent the recreational activities of medieval monks, with such productions as the *Life of Christ* made up from lines extracted from the Homeric poems or Virgil.<sup>3</sup>

Another use of texts which is more recreational than cryptographic is the formation of acrostics, where the secret information is concealed, frequently by the use of initial letters in a specially prepared text. The *Hypnerotomachia Poliphili*, where a Dominican monk confesses his love for one Polia,<sup>4</sup> and certain of the poems of Edgar Allan Poe are good examples.<sup>5</sup>

Anagrams, which use the letters of one text to form a secret message were frequently used by 17th century scientists to conceal and establish priority of discoveries. Unfortunately, it is often possible to anagram several perfectly good plain texts from a particular collection of letters, as witness the several thousand anagrams made up from the angelic salutation "Ave Maria...."<sup>6</sup> An anagram can be considered as an unkeyed transposition cipher and is worthless for the purposes to which it was put by Galileo, Huygens, and others.<sup>7</sup> Oddly, anagramming is often used as a last resort, especially by the unskilled, to unscramble impossible cryptograms even when there is no logical basis whatsoever for doing so.

## THE SIXTEENTH CENTURY

A rather impractical method of book cipher was suggested by Blaise de Vigenere in 1586, which consists of placing a transparent sheet over the pages of a book and underlining the words you wish to use on the transparency. When a copy is sent to the receiver, he can place it over his copy of the book and see which words have been marked.<sup>8</sup> (This is really the equivalent of making a simple grill by cutting holes in a sheet of paper and sending it to the receiver to fit over a page.)<sup>9</sup> But the chances of finding the desired words for a message in a page or so of print is very slim.

Vigenere also describes the book cipher method which is probably the one most commonly used--that in which a letter is indicated by using numbers to show page, line, and location of the letter within the line.<sup>10</sup> This obviously takes a good many figures to encipher a single letter. He feels that such a cipher would be impossible to break without the key, but that, nevertheless, it is slow and tiresome to use and subject to error. He ascribes the method to Leone Battista Alberti of Florence (d. 1472) and quotes Alberti as saying that the method was "worthy of an emperor or a king." However, this is apparently based on a mistranslation. In actuality Alberti was referring to his own invention of the disk cipher and a more correct translation of the passage in Vigenere would be "It is sufficient to have mentioned it (the book cipher) in passing, because I have seen some who prize it very highly, just as a certain Leone Alberti of Florence prizes his own cipher (the disk cipher), 'worthy,' he says, 'of an emperor or king.'"<sup>11</sup>

The first actual description of a true book cipher that the authors were able to discover is that of Jacobus Silvestri in 1526.<sup>12</sup> Silvestri writes of a sort of code book, or dictionary, with root words in multiple columns. A unique symbol is associated with each column and row, and additional symbols are available for grammatical inflections. This is really an early form of an artificial language. Some current investigators are of the opinion that the famous Voynich manuscript may be composed in a form of artificial language.<sup>13 14</sup> The Voynich Manuscript is often referred to as the "most mysterious manuscript in the world." Even its authorship is a puzzle. It has been attributed by some to Roger Bacon in the 13th century and has been thought to contain early records of scientific discoveries. In actuality, not even its date of origin or the language it is written in has ever been satisfactorily established.

Another early practitioner of the book cipher was the famous mathematician Girolamo Cardano, writing in the 1550's, whose impractical method was to find the words he wanted in a text and then write and send the words before or after the desired ones, altering them to make connected sense and adding, if necessary, extra words in parentheses. It is easy to imagine the recipient of such a cipher searching through his copy of the book which was used looking for the words before and after those in the cipher. Charles J. Mendelsohn charitably says "Other ways of communicating with the aid of two copies of the same book have been devised, but this one has never come into favor."<sup>15</sup>

A similar method of using a book was proposed by Giovanni Battista Porta in 1563,<sup>16</sup> but by 1586 Vigenere was proposing several ways of using a book such as the oiled-paper transparency already referred to and the commonly used method of using numbers for page, line, and location of the letter within the line, but also of disguising the cipher as an astronomical table. The signs of the zodiac were used to indicate the page, the numbers in the degree column to indicate the line, and the numbers in the minute column to indicate the individual letter. As he says, a mathematician would be suspicious if he checked the numbers closely.<sup>17</sup>

An aberration in the story of book ciphers is the famous biliteral cipher of Francis Bacon<sup>18</sup>, which has caused so much error and confusion in the Bacon-Shakespeare authorship question. With a biliteral cipher, the text of a book is printed in two fonts, an A-font and a B-font, and the secret message is encoded in 5-letter groups such that aaaaa=A, aaaab=B, aaaba=C, etc. Bacon may indeed be the first to have employed a binary numbering system. It is a perfectly legitimate cipher, but depending as it does on often questionable differences in type fonts, has frequently been misused or misunderstood. The Great Cryptogram of Ignatius Donnelly<sup>19</sup>, intended to expose Francis Bacon's Cipher in the Shakespeare plays, was nothing more than an elaborate cipher unwittingly constructed in reverse by Donnelly that allowed him to recover a desired plain text. The vivid imaginings of Elizabeth Wells Gallup come to mind also.<sup>20</sup>

#### THE SEVENTEENTH CENTURY

The 17th century is marked by the often naive efforts of scientists to protect their claims to priority in discoveries by concealing them in anagrams. Huygens and Galileo furnish good examples. They didn't realize that an anagram is nothing more than an unkeyed transposition cipher containing many legitimate solutions. Galileo's discovery of the phases of the planet Venus "*Cynthiae figuras aemulatur mater amorum*" (The mother of love (Venus) imitates the phases of Cynthia (the moon)) was concealed in the anagram "*Haec immatura a me jam frustra leguntur o.y.*" (These unripe things are now read by me in vain o.y.).<sup>21</sup>

#### THE EIGHTEENTH CENTURY

By the 18th century Christian Breithaupt writes of using numbers for the page and line in which the desired letter occurs first. But he



felt the method was not secure, since a clever investigator might find out what key book had been used.<sup>22</sup> Moreover, a great deal of searching might have to be done to find the desired letter as the initial letter of a line. Nevertheless, he says the method has become very common and is known to wanderers and beggars.

Later in the century Philip Thicknesse writes of a method of numbering the pages, lines, and words of a text.<sup>23</sup> He says that it is "scarce possible to be disclosed without the key." But searching through many pages to find a desired word would make this a very slow system for practical use.

With the coming of the American Revolution a number of different and improved book ciphers made their appearance. Numbering schemes were adopted that permitted the enciphering of individual letters. This made them somewhat easier to use and more precise. An early example of this sort was sent to Benjamin Franklin in Paris on 10 June 1776 by Barbeau-Dubourg, the translator of Franklin's works.<sup>24</sup>

Barbeau-Dubourg proposed numbering each letter of a key text, whose source has recently been found. With the assistance of Dr. Eric Gans, Chairman of the French Department, University of California at Los Angeles, Dr. Leighton was able to extend the Barbeau-Dubourg key somewhat. As printed in the Franklin Papers (Vol. 22, p. 470) the short example of plain and cipher text was:

3	2	19	5	23	16	12	44	53	10	51	4	61	36	17	6	24	71	1		
M	A	F	E	M	M	E	E	T	D	E	U	X	F	I	L	L	E	S		
42	28	37	33		82	54	11	9	8	47	59	88	13	69	31	92	72	34	56	73
V	O	U	S																	
6	94	4	20	40	100	68	48													

from which the fragmentary key:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	...
S	A	M	U	E	L				D		E				M	I		F	

was derived. Dr. Gans suggested completing the plain text phrase "Ma femme et deux filles" with the words "embrassent de tout leur etre" to form the greeting "My wife and two daughters embrace you with all their being." This extends the key to:

S A M U E L \_ A R D B E N \_ \_ M I \_ F R \_ \_ M L \_ \_

plus other scattered letters and indicates that the key begins with the names Samuel Ward and Benjamin Franklin (with an M mistakenly used for K) and suggests that the key was made up of names taken from the membership of the Secret Committee of the Continental Congress, which

was obviously in touch with Barbeau-Dubourg. (The Secret Committee was the predecessor of the present day State Department.)

Another of Franklin's correspondents was C.W.F. Dumas, who used a similar method of cipher.<sup>25</sup> By carefully following clues in Dumas' letters, Dr. Matyas determined that the book Droit des Gens was used as the key.<sup>26</sup> The key was mentioned subsequently in the Franklin Papers, indicating that it was found by at least one other person as well.<sup>27</sup>

One of the leading British generals in the Revolution, Frederick Haldimand, Governor of Canada, used a book cipher whose key was discovered some years ago "by an officer to whom Wm. F. Friedman (head of the U.S. Army cryptanalytic effort before and during WWII) was indebted."<sup>28</sup> It turned out to be the title page of a British Army List, a small book sure to be in the hands of every officer.<sup>29</sup> Further analyzing the problem, Dr. Leighton was recently able to extend this solution by finding the system which governed the associated code list. In his enciphered correspondence Haldimand was using numbers of the form 10-19 to stand for complete words. By studying the numbers and associated words found in some examples of Haldimand's deciphered messages,<sup>30</sup> Dr. Leighton was able to determine that the second part of the number stood for the letter which began the word and the first part indicated its position in the code list under that letter. Thus the word "there", represented by number 10-19, is the 10th word in the 19th sublist (of words all beginning with the letter "T"). Enough numbers and words have been analyzed to suggest that the code list had forty to fifty words per sublist.

Benedict Arnold tried to use Blackstone's Commentaries as a key book, but found it so slow and cumbersome that he and Major Andre soon switched to Bailey's Dictionary, 21st edition, 1770, using a system based on page, column, and line numbers with a plus one displacement. Thus, "Zoroaster" became 928.2.2 not 927.1.1. If they had switched key books earlier, West Point might have been lost to the British. Two other famous British generals, Cornwallis and Clinton, in 1781, based a code on the 1777 edition of Entick's Dictionary, while Aaron Burr, in 1805, used a similar method based on the 1800 edition of the same book.<sup>31</sup>

## THE NINETEENTH CENTURY

The first commercial venture to bring cryptography to the fingertips of most Americans occurred in 1805 with the publication of a small dictionary "to enable any two persons to maintain a

correspondence, with a secrecy, which is impossible for any other person to discover."<sup>32</sup> A short list of directions for using the dictionary and numbering the words in a pair of books was provided. But there is nothing to show that this dictionary code was well-received, perhaps indicating that it was published more as a marketing ploy, since two books were required to make it work.

Kluber's Kryptographik, published in 1809, is the most complete, useful, and possibly most influential of the early how-to books on cryptology.<sup>33</sup> He gives his highest recommendation to book ciphers as being extremely secure. Provided also are instructions showing how words, syllables, and individual letters can be enciphered differently each time they occur. For example, with a key text where figures 3 and 61 stand for "also" and "sich", the cipher 3.61... denotes the plain text "ach", that is, the first letter of "also" and the third and fourth letters of "sich."<sup>34</sup> (Even in this simple example the letter "s" could be enciphered in two ways using 61. or 3...)

A cipher from about 1810 found in the Thurn und Taxis archives in Germany, by Dr. Erich Huttenhain, uses page, line, and letter numbers.<sup>35</sup> He was able to recover a few words of the key, but has not yet found the key book.

The most sought after solution to a book cipher is that of the Beale ciphers, alleged to contain instructions to the Beale Treasure, supposedly buried in Virginia about the 1820s.<sup>36</sup> Of the three ciphers, the solution to one has been known about 100 years. It is based on numbering each word in the Declaration of Independence (When 1 in 2 the 3 course 4 ....) and then using only the initial letters of the words: 1=W, 2=I, 3=T, 4=C, etc. Despite the best efforts of many, including the Beale Cypher Society, no key document or book breaking the other two ciphers has been found.

Better luck has been achieved with the book ciphers of Nicholas Trist, whose unauthorized negotiations with Mexico ended the Mexican War in 1847 and greatly expanded U.S. territory. While in Mexico, he found it necessary to devise two ciphers to permit secure communication with his superiors in Washington. One was based on numbering each letter of a passage; the other on a triple coordinate system in which individual letters were designated by numbers corresponding to the page, line, and position of the letter within the line. He described the key book circuitously in letters to James Buchanan, the Secretary of State, later to become President. In a model of scientific and literary detections, described in a recent issue of *Cryptologia*, Dr.

Matyas, by carefully delimiting the field of investigation and then checking methodically hundreds of books, identified the key book, thereby unlocking additional enciphered Trist material.<sup>37</sup>

For anyone who wishes to undertake a similar quest, here is a good opportunity. Jefferson Davis, President of the Confederacy, proposed using a dictionary as a code book.<sup>38</sup> All that is necessary is to find a three column dictionary in which the 20th word in the left column of page 146 is "Junction."

The use of a dictionary as a key book was particularly common in Victorian times and was frequently proposed in popular magazines.<sup>39</sup> This really results, as previously said, in a one-part code with page numbers increasing from A to Z. Without super-encipherment of the numbers it is soon possible to estimate the size of the dictionary, number of columns, and average number of words per column.

Perhaps the most curious use of a dictionary is in an international astronomical cipher code published at Harvard College Observatory in 1881.<sup>40</sup> Here we have a reverse dictionary code in which words from a particular edition of a dictionary are used to represent numbers. As an example, in this astronomical cipher, the first word always represents the day of the year and the time of day; the second word indicates the "distance of perihelion from node" and the third word stands for the "longitude of node." The message "Customably digitated butternut" is deciphered in this way: On page 136 the 73rd word "customably" = the 136th day of the year (16th of May), 73 is the time of day expressed decimally. "Digitated" the 8th word on page 150 = 150°8', the distance of perihelion from node, and "butternut" the 28th word on page 91 represents 91°28", the longitude of node.

Sherlock Holmes, never unaware of current trends, used a book cipher based on a non-existent edition of Whitaker's Almanac in The Valley of Fear.<sup>41</sup> His creator, Sir Arthur Conan Doyle, sent messages to British POWs in Germany in World War I by inconspicuous pinpricks under desired letters, a method already described.<sup>42</sup>

## THE TWENTIETH CENTURY

The outstanding achievement in solving a book cipher must go to William F. Friedman, who cryptanalyzed a Hindu book cipher in World War I. He did so without knowledge of the key book, although the book was later identified.<sup>43</sup>

Herbert O. Yardley, author of The American Black Chamber, worked in China shortly before our entry into World War II. Some of his achievements are told in his recently published The Chinese Black

Chamber, including the use of a book as a key to encipher messages in the Chinese Telegraph Code (Ming Code).<sup>44</sup> After receiving a message, which began with the 5-letter group EHEER, he theorized that the letters might indicate message numbers and date. Thus, he equated EHEER = 10112 = message 101 of the 12th. A similar equivalency was made on successive days, which allowed him, after putting the letters in normal order, to recover parts of three words:

```

0 1 2 3 4
H E R
L I G H
G R   I N

```

Investigation revealed that the words came from pages 17, 18, and 19 of Pearl Buck's The Good Earth.

Many of the systems using books result in cipher messages which are much longer than the original plain text. A way to avoid this is the running key, where a text from a book is combined in one of many possible ways with the desired plain text to produce a cipher text of the same length. Robert Graves, in his I, Claudius (a fictional work), describes a running key cipher in which the key is the first hundred lines of Homer's Iliad.<sup>45</sup> Each letter in the cipher is represented by the number of letters in the alphabet (ABC...Z) between it and the corresponding letter in Homer. Suppose the Iliad began with the word "Achilles," then the recipient of the cipher 10 6 4 3 would count 10 letters from A, 6 from C, 4 from H and 3 from I and would find the plain text KILL:

```

      A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
10  A - - - - - - - - - K
6   C - - - - - I
4   H - - - L
3   I - - L

```

Another common way of using a running key is illustrated:

```

P | W H E N I N T
K | T A K E O F F
C | X T G R G S M

```

where P = plain text, K = Key, and C = Cipher text. When two standard alphabets are slid against each other with the "T" of the second alphabet aligned under the "W" of the first alphabet, in this way

```

ABCDEF GHIKLMNOPQRSTU V|W|XYZABCDEFGHIJ...
ABCDEF GHIKLMNOPQRS|T|UVWXYZ

```

then the cipher "X" is the letter in the second alphabet aligned under the letter "A" of the first alphabet, assuming that the first alphabet is continuous.

William F. Friedman, who broke the Japanese diplomatic cipher used in World War II, was the first person known to the authors to systematically solve running key ciphers.<sup>46</sup> His basic approach was to assume that both the key and the plain text were intelligible sequences. Once started, an extension of the key automatically extends the plain text and vice versa. The example above can illustrate the principle. If the first two words "When" (plaintext) and "Take" (key) are known, the assumption of the word "in" (plain text) would automatically give "of" for the key. Completion of the word to "off" would give "t" as the beginning of the next word in the plain text suggesting "the". Thus the key and plain text can be worked against each other to extend both.

If the key is not intelligible, but random (e.g. using a book of random numbers or letters), then one has a completely unbreakable one-time system. But no part of the key can be used more than once, so that the amount of keying material needed must be equal to the anticipated amount of plain text to be communicated, which could be substantial. A code book of random numbers or letters is not easily hidden among the books of an ordinary library, so that it may be difficult to maintain the secrecy of the keying material without drawing undue attention to it. An example of a one-time system is found in one of Che Guevara's worksheets, which was discovered after his death in 1967. The worksheet is in the following form:

```

P  72 8 32 34 8
K  34 6 51 42 1
-----
C  06 4 83 76 9

```

The plain line (P) represents a simple numerical substitution of the plain text LECHE (i.e. Fidel Castro). The key (K) is a stream of random numbers. The cipher text (C) is produced by non-carrying addition.<sup>47</sup>

## CONCLUSION

We have traced the origin and development of book ciphers over a period of at least 400 years and seen a few of the ways in which a book may be used for secure communications. The Beale ciphers are excellent proof that sometimes book ciphers are very secure indeed!

## ACKNOWLEDGEMENT

Dr. Leighton would like to thank the Center for Medieval and Renaissance Studies at the University of California, Los Angeles, for making some of his research possible.

## FOOTNOTES

1. See example in Sir Arthur Conan Doyle "The Hound of the Baskervilles", The Complete Sherlock Holmes, Garden City, N.Y., n.d., p. 801.
2. Aeneas Tacticus. 31.1-3. Albert C. Leighton, "Secret Communication among the Greeks and Romans," Technology and Culture, Vol. 10, No. 2, April 1969, p. 149.
3. Article "Cento," Encyclopaedia Britannica, 11th ed., New York, 1910, p. 674.
4. David Kahn, The Codebreakers, New York, 1967, pp. 873-4.
5. William F. and Elizebeth S. Friedman, The Shakespearean Ciphers Examined, Cambridge, 1957, p. 97.
6. Kahn, Codebreakers, p. 869.
7. Kahn, Codebreakers, p. 773.
8. Blaise de Vigenere, Traicte des Chiffres ou Secretes Manieres d'Ecrire, Paris, 1586, pp. 208-9.
9. The grill cipher had already been invented by Girolamo Cardano in the 1550s. See Charles J. Mendelsohn, "Cardano on Cryptography," Scripta Mathematica, Vol. 6, 1939, pp. 164-5.
10. Vigenere, Traicte, pp. 208-9.
11. Charles J. Mendelsohn, "Blaise de Vigenere and the 'Chiffre Carre'," Proceedings of the American Philosophical Society, Vol. 82, No. 2, March 1940, pp. 117-8.
12. Jacobus Silvestri, Opus Novum, praefectis arcium, Rome, 1526.
13. Philip M. Arnold, "An Apology for Jacopo Silvestri," Cryptologia, Vol. 4, No. 2, April 1980, pp. 96-103.

14. Mary E. D'Imperio, The Voynich Manuscript, National Security Agency, Ft. Meade, Maryland, 1978, pp. 67-8, 118.
15. Mendelsohn, "Cardano on Cryptography," pp. 161-2.
16. Giovanni Battista Porta, De Furtivis Literarum Notis, vulgo de Ziferis libri IV, Naples, 1563, pp. 106-7.
17. Vigenere, Traicte, pp. 208-9.
18. Francis Bacon, Tyvoe bookes of Francis Bacon of the Proficiencie and Advancement of Learning, Divine and Humane, London, 1605, p. 61.
19. Ignatius Donnelly, The Great Cryptogram, Chicago, 1888.
20. On this and other pseudo-ciphers see Friedman and Friedman, Shakespearean Ciphers.
21. Kahn, Codebreakers, p. 773.
22. Christian Breithaupt, Ars deciffratoria sive scientia occultas scripturas solvendi et legendi, Helmstadt, 1737, p. 20-21.
23. Philip Thicknesse, A treatise on the art of decyphering and of writing in cypher, London, 1772, pp. 111-12.
24. Franklin Papers, Philadelphia, 1982, Vol. 22, p. 470.
25. National Archives Microfilm Publications, Microcopy no. 247, Papers of the Continental Congress, 1774-1789, Roll 121, Item no. 93, Letters Received from Charles W. F. Dumas. Washington, 1959.
26. Le Droit des Gens par M. de Vattel, Amsterdam, 1775, pp. iii-v.
27. Franklin Papers, Vol. 22, p. 464.
28. William F. Friedman, Six Lectures on Cryptology, National Security Agency, Ft. Meade, Maryland, 1963, pp. 41-3.
29. A List of the General and Field Officers as they rank in the Army, London, 1778.
30. Public Archives Canada, Haldimand Papers MG 21, Additional Manuscripts 21807 and 21808, Microfilm Roll A-741, Correspondence with Sir Henry Clinton and others at New York, 1777-1783.
31. Carl van Doren, Secret History of the American Revolution, New York, 1969, pp. 200, 204. Kahn, Codebreakers, pp. 177-8, 183, 186.
32. A Dictionary, to enable any two persons to maintain a correspondence, with a secrecy, which is impossible for any other person to discover, Hartford, 1805.
33. Johann Ludwig Kluber, Kryptographik, Tübingen, 1809.
34. Kluber, Kryptographik, p. 349.
35. Albert C. Leighton, "Ciphers in Bavarian Archives," Proceedings of the Second Beale Cypher Symposium, Washington, 1979, p. 79.
36. James B. Ward, The Beale Papers, Lynchburg, 1885.



37. Albert C. Leighton and Stephen M. Matyas, "The Search for the key book to Nicholas Trist's book ciphers," Cryptologia, Vol. 7, No. 4, October 1983, pp. 297-314.
38. Friedman Lectures, p. 75.
39. Examples of dictionary codes are found in G.P.B. (George Parker Bidder), "Ciphers and Cipher-Writing," Macmillan's Magazine, Vol. 23, 1871, p. 330; (Anonymous) "Missives in Masquerade," Cornhill Magazine, Vol. 29, 1873, p. 179; (Anonymous) "The Art of Secret Writing," The Practical Magazine, Vol. I, 1873, p. 318; (Anonymous) "Writing to Conceal One's Thoughts," All the Year Round (conducted by Charles Dickens), Vol. 35, 1875, p. 510; T.J.A. Freeman, "Cipher," American Catholic Quarterly Review, Vol. 18, 1893, pp. 863-4; George Wilkes, "Cryptography," Cosmopolitan, Vol. 36, 1903, p. 716.
40. William W. Payne, "Astronomical Cipher Messages," The Sidereal Messenger, Vol. 4, No. 6, pp. 161-3.
41. Arthur Conan Doyle, "The Valley of Fear," Complete Sherlock Holmes, p. 904.
42. Arthur Conan Doyle, Complete Sherlock Holmes, Preface, p. x.
43. Kahn, Codebreakers, p. 371-3.
44. Herbert O. Yardley, The Chinese Black Chamber, Boston, 1933, pp. 114-24, 129.
45. Robert Graves, I, Claudius, New York, 1934, Chapter 17, pp. 232-33.
46. Kahn, Codebreakers, p. 375. William F. Friedman, Methods for the Solution of Running-Key Ciphers, Riverbank Publication, No. 16, Geneva, Illinois, 1918.
47. David Kahn, Kahn on Codes, New York, 1983, pp. 139-44. Barbara Harris of New York solved the simple numerical substitution for the plain text with the following result:

```

      8 2 0 6 4 9 1
      e s t a d o y
    3 b c f g h i j
    7 k l m n n p q
    5 r u v w x z

```

Note that it was not the one-time system that was cryptanalyzed but only the simple substitution of the worksheet.

AN UPDATE ON FACTORIZATION AT  
SANDIA NATIONAL LABORATORIES\*

J. A. Davis and D. B. Holdridge

Sandia National Laboratories  
Albuquerque, New Mexico 87185

ABSTRACT

Since Crypto 83 we have had considerably more experience in factoring large integers. Implementation of various modifications to the quadratic sieve algorithm have enabled the factorization of hard 70-digit numbers in times comparable to 50 digits one year ago. These modifications include:

- 1) Subsequences with large divisors (Special q's).
- 2) Multipliers to improve quadratic properties.
- 3) Increased size of prime base using segmented Gaussian Elimination.
- 4) Optimization of the code with respect to Cray hardware.

Using this code in its various stages of development the 10 most wanted numbers from the Cunningham Project have been factored. Details will be published elsewhere.

---

\* This work was performed at Sandia National Laboratories supported by the U. S. Department of Energy under Contract No. DE-AC04-76DP00789.

# An LSI DIGITAL ENCRYPTION PROCESSOR (DEP)

R. C. Fairfield, A. Matusevich, and J. Plany

AT&T Bell Laboratories  
25 Lindsley Drive  
Morristown, N.J. 07960

## ABSTRACT

This paper describes an LSI digital encryption processor (DEP) for data ciphering. The DEP combines a fast hardware implementation of the Data Encryption Standard (DES) published by the National Bureau of Standards (NBS) with a set of multiplexers and registers under the control of a user programmed sequencer. This architecture enables the user to program any of the DES modes of operation published by NBS. In addition, multiple ciphering operations and multiplexed ciphering operations using up to four different keys may be programmed and internally executed without any external hardware.

The DEP is designed as a standard microprocessor peripheral. This LSI device should reduce the current cost and simplify the process of encrypting digital data to a point where it is feasible to include a ciphering function in modems, terminals, and work stations. The ability to internally program cascaded ciphers should substantially increase the security of the DES algorithm and hence, the life of the encryption equipment.

## INTRODUCTION

In January 1977 the National Bureau of Standards (NBS) adopted an IBM developed block cipher called the Data Encryption Standard (DES) [1]. Approximately four years later, in December 1980, the NBS published a follow-up document titled "DES Modes of Operation" [2] which describes four DES operating modes and some of their

characteristics. This paper describes a new LSI device called the Digital Encryption Processor (DEP) developed by AT&T Bell Laboratories and manufactured by AT&T Technologies. The DEP has been validated by the NBS as complying with the DES. Other devices have also been certified. This, however, is the first LSI device to incorporate all of the standard DES modes of operation into a single integrated circuit and provide the user with the flexibility to program unique or custom ciphering functions.

Unless provisions are made now to effectively lengthen the key space of current DES enciphering modes, the encryption equipment life may be shortened by integrated circuit technology advances. To date, the best known attack on the DES algorithm is a brute force search of the key space under a known plaintext attack. Today's fastest DES integrated circuits can perform a maximum of 250K ciphering operations/second. The operating speed may be increased by a factor of 12 by shrinking the design rules. Further performance improvements may be achieved by pipelining the DES algorithm (factor of 16). Pipelining would require sixteen sets of: (1) 64 bit L and R registers; (2) 56 bit C and D registers; (3) 32 "exclusive-or" gates ; (4) 48 "exclusive-or" gates; and (5) 8 s-boxes (2044 bits of ROM). This would increase the integrated circuit transistor count by approximately a factor of 16. Certainly, this would be a huge integrated circuit even by todays standards. Despite these speed improvements, an array of devices would still be required to search the key space in a reasonable amount of time. Therefore, each device should be capable of independent operation. This would require each integrated circuit to have an independent controller, a comparator function to flag matching ciphertext, and a read/write key counter register. Then, using an array of five-hundred devices each independently searching a different part of the key space, all possible keys could be checked in one month. This is a distressing result for a DES device or board manufacturer who would like to see a product in the field for a period of years. For this reason the DEP device may be programmed to internally perform cascaded ciphering operations using up to four different keys. A cascade of k ciphers may not be equivalent to increasing the key size (56 bits) by a factor of k since a time-memory tradeoff may be made; however, it is certainly far more work than searching the key space of a single cipher. See reference [3]. S. Even and O. Goldreich have shown that

a cascade of two DES's can be cracked in time  $2^{71}$  and space  $2^{41}$  [4]. Work increases on this order would extend the practical life of the DES for years. It may also be possible to rearrange the feedback in a cascaded cipher to prevent a meet in the middle known plaintext attack.

The DEP combines a fast hardware implementation of the DES with a sixty-four bit input shift register, a sixty-four bit output shift register, a set of multiplexers necessary to configure the operating modes, a data latch, and four sets of key and initial value registers. Control over this hardware is provided by a user programmed sequencer. This sequencer provides the flexibility necessary to program any of the four DES operating modes and to tailor the encryption function to the system requirements. Additionally, the four different key and initial value registers may be used to program multiplexed ciphering operations or to provide for enhanced security requirements by programming multiple ciphering operations using different keys.

The DEP is designed as a microprocessor peripheral and is packaged in a standard forty pin dual-in-line package. Figure 1 shows a block diagram of the device. There are two separate parallel bidirectional eight bit ports, two separate serial bidirectional data ports and a serial key port. All of these ports may be read or written asynchronously with respect to the clock input. The separate data ports are provided to increase data throughput and security by allowing separate plaintext and ciphertext buses. There are seven possible data port configurations. The serial key input port would typically be used to load a key from external circuitry, say a ROM, that the user keeps locked up when not in use. Microprocessor polled or interrupt systems may be configured, since output flags may be read from the data buses or on independent output pins. Maximum data rate for the device in any of the standard operating modes is as follows:

Input Clock (Tc)	2.5 MHz (worst case)	4.5 MHz (nominal)
	-40 to 80 deg C	
Instruction Period	$2 \cdot T_c$	
Ciphering op/sec	73.5K (worst case)	132K (nominal)

All four NBS defined operating modes may be executed in a minimum of 17 instructions. If the entire DES output block (64 bits) is used for the ciphering operation, the worst case data throughput is 0.59 megabytes per second.

The internal user programmed sequencer enables the device to

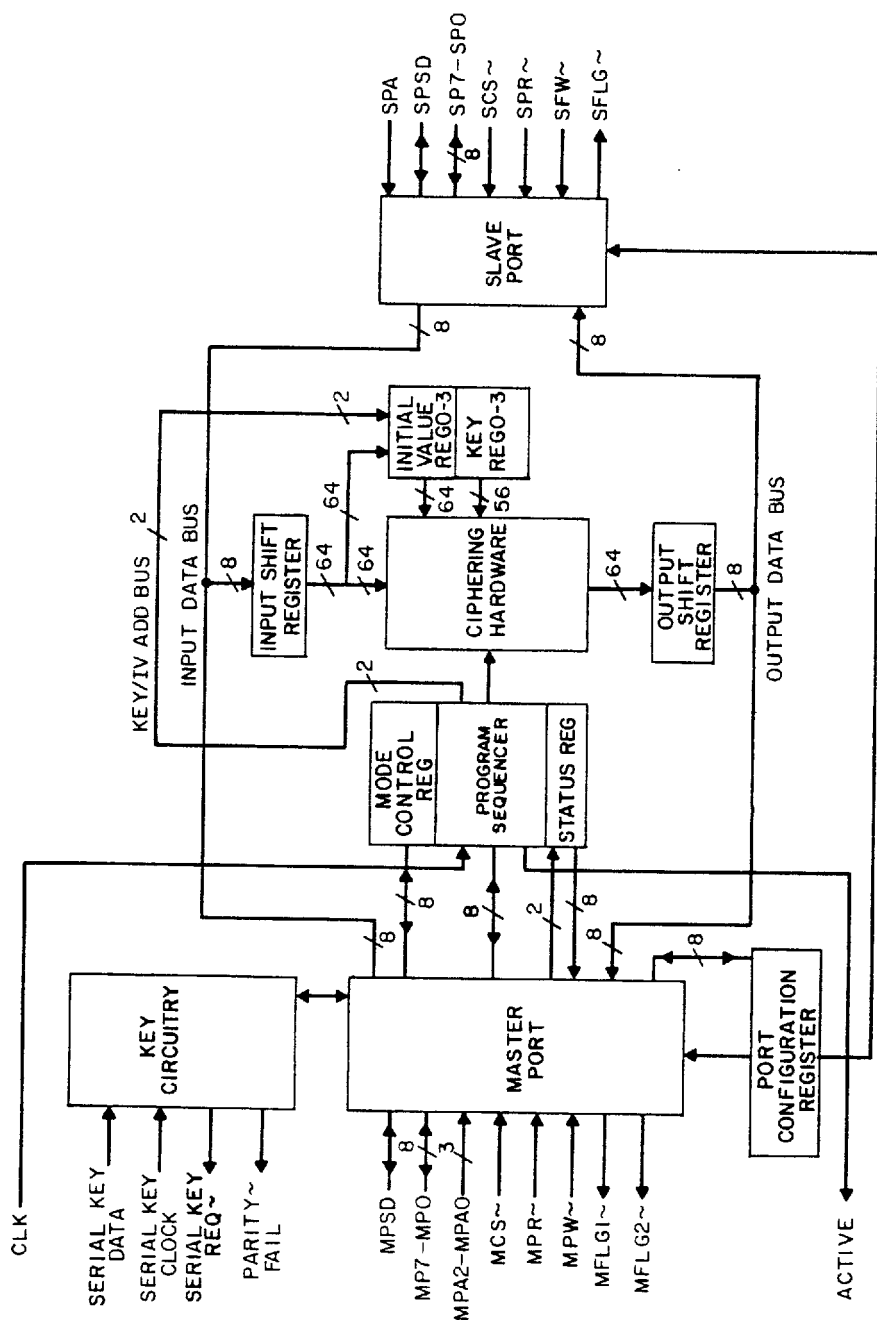


FIGURE 1  
DEP BLOCK DIAGRAM

accommodate special system requirements and reduces host processor overhead. The DEP should reduce the cost and simplify the process of encrypting digital data to a point where it is possible to include a ciphering function in modems, terminals and work stations. In addition, the ability to internally program cascaded ciphers should substantially increase the security of the DES algorithm. This paper describes the DEP architecture, the micro-code instruction set, and then gives some unique applications.

## ARCHITECTURE

The DEP architecture may be divided into two sections: the ciphering hardware and the user programmed sequencer.

### THE CIPHERING HARDWARE

The DES specifies a cryptographic algorithm which is a nonlinear sixty-four bit block cipher using a fifty-six bit key. The components of the algorithm are simple and individually weak. They consist of permutations, combinations ("exclusive-or" sums) of the data and internal key bits, and nonlinear substitutions. These weak elements are combined and the data are encrypted in sixteen iterations through them. Sixteen 48 bit internal keys are generated by rotating and permuting the 56 bit DES input key. Although NBS has published the DES [1], no cryptographic analysis or justification for the specific elements in the algorithm has been published. The published literature does, however, provide some insight into the inner workings of the algorithm [5] and [6]. The key input to the DEP device is a 64 bit number with the least significant bit in each byte, or every eighth bit in a serial key load, a parity bit. Odd parity is checked and a flag is set if parity fails. Device operation is not inhibited by a parity failure.

Figure 2 shows a block diagram of the DEP ciphering hardware, with the DES key schedule and enciphering circuitry enclosed in dotted lines. The algorithm specified in the DES was designed to be implemented in hardware (not software). There are several permutation matrices specified in the standard and the penalty for a software implementation is an inordinate amount of time spent shuffling or permuting bits. This operation has no overhead in hardware, since the

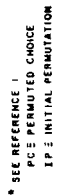


FIGURE 2 DEP CIPHERING HARDWARE AND PERIPHERAL CIRCUITRY



permutation matrix is simply a crisscross of wires. The DES section of the enciphering circuitry consists of: a 2:1 multiplexer with 64 sections; two 32 bit L and R registers; 32 "exclusive-or" gates; and a cipher function,  $F$ , of the internal key and R register. Figure 3 illustrates the  $F$  function. The eight s-boxes shown are the nonlinear algorithm elements and are implemented as eight ROMs, each consisting of 64, four bit words (six address lines and four outputs).

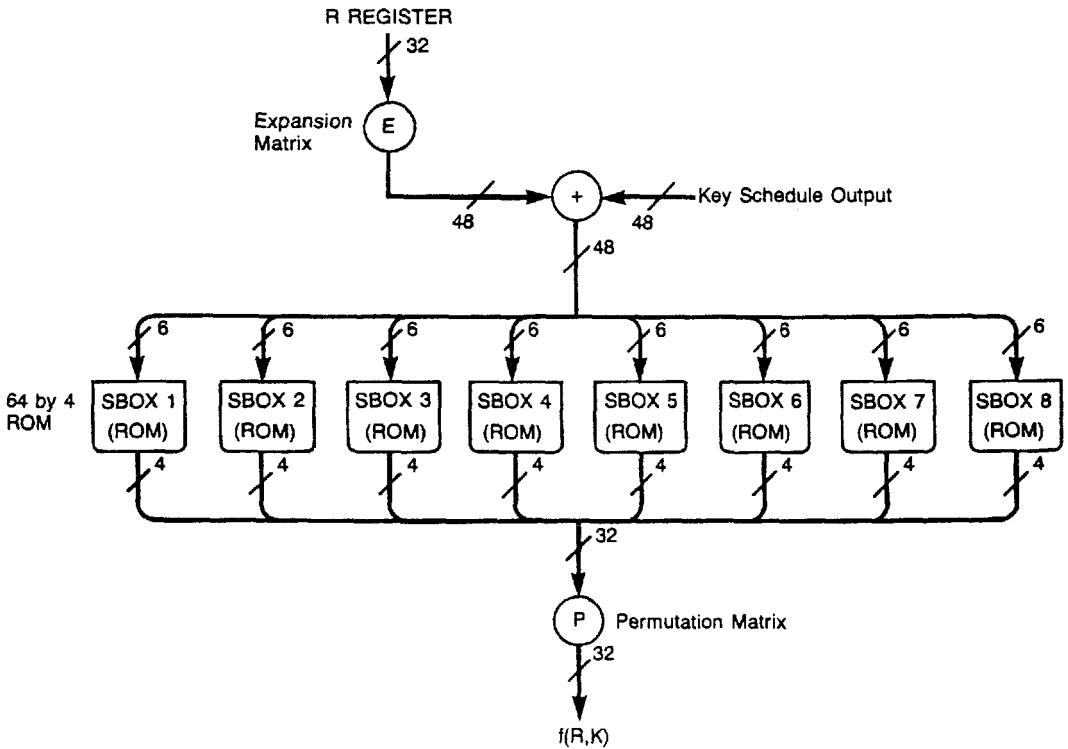


FIGURE 3.  $F$  FUNCTION

Since the DES algorithm is a block cipher, there is a one to one mapping of the input block to the output block. To a cryptographer, it is disconcerting to know that a recurring plaintext block will duplicate the earlier ciphertext block. This leaves the crypto system

vulnerable to traffic analysis and the possibility of insertion or deletion of messages by an active intruder. Hence, the NBS published the "DES modes of operation" [2]. Four modes are defined:

1. Electronic Codebook (ECB) is a straightforward implementation of the DES algorithm.
2. Cipher Block Chaining (CBC). To begin the operation, an initial value is added modulo 2 to the first plaintext block to form the DES input block. The DES output is the ciphertext. This output is fed back and added modulo 2 to the next plaintext block forming the new DES input block. CBC produces a ciphertext dependent on previous plaintext blocks.
3. K-bit Cipher Feedback (CFB). Starting with an initial value as the DES input block, K plaintext input bits are added modulo 2 to the K most significant bits in the DES output block. The result is the K-bit ciphertext which is fed back and shifted into the K least significant bits of the DES input block to form the next DES input block.
4. Output Feedback (OFB). Starting with an initial value, the DES is operated as a pseudo random bit stream generator (the DES output is fed back as the input). Ciphertext is produced by adding the plaintext to the random bit stream modulo 2.

Figure 2 shows the sets of multiplexers, the "exclusive-or" gates, and the data latch necessary to configure the DES operating modes. MUX 6 and the latch register are used to shift the input data block for the CFB mode. MUX 13 is used to select the input to initial value registers 0 through 3. The initial value registers may be used to hold temporary products in a multiple encryption operation, or to store the next DES input block for the current ciphering operation, before jumping to a different ciphering operation. The input and output shift register circuitry is clocked by the rising edge of the decoded data write and read strobes applied to the chip. When the input shift register is filled, an ISRFULL flag is set and the DEP can cipher the new data and clear the flag. When the output shift register is empty, an OSREEMPTY flag is set and the DEP can reload this register and clear the flag. These two flags may be read by the user on either of the two eight bit data ports or on separate output pins. This structure allows the external read and write strobes to be independent of the DEP clock. To achieve maximum data throughput a

user would have to complete the reading and writing of data during the DEP ciphering operation.

#### THE USER PROGRAMMED SEQUENCER

The two eight bit bidirectional data ports, master and slave, may be thought of as plaintext and ciphertext ports, respectively. All control registers must be written through the master port. Other than data, which may be read or written for ciphering, only three flag bits of the status register may be read from the slave port. See Tables 1 and 2. Control over the ciphering hardware is provided by the user programmed sequencer. A block diagram is shown in Figure 4. The sequencer executes a 22 bit instruction every two clock cycles. Depending on the address in the program counter, these instructions may come from either a RAM or ROM program memory.

The ROM contains three programs and one subroutine. The subroutine executes the DES algorithm using whatever key is currently in the C and D registers (Key Schedule, Figure 2) to encipher whatever data is sitting at the input to the initial permutation matrix (labeled IP, DES Enciphering circuitry). There are four pairs of key and initial value registers that may be externally loaded. These registers are loaded by writing the address (0 through 3) of the key/initial value pair to an internal status register. Then the appropriate ROM program is executed. The three programs are described (ROM Code, Table 3):

1. A load initial value program waits for an eight byte number to be written to the master port. When the ISRFULL flag is set, this number is clocked into the initial value register addressed by the status register.
2. A load key program waits for an eight byte number to be written to the master port. When the ISRFULL flag is set, this number is clocked into the key register addressed by the status register. Odd parity of each byte is checked. The least significant bit in each byte is the parity bit.
3. A serial load key program waits for a sixty-four bit number to be clocked into the serial key data port using the serial key clock. When this program is executed, a hardware key request pin goes active. When the key is loaded into the input shift register, the sequencer clocks the number into the key register

MASTER PORT ADDRESS	REGISTER	READ OR WRITE	CONTENT								LSB
			B7	B6	B5	B4	B3	B2	B1	B0	
0	INPUT SHIFT REGISTER (ISR)	W	DI1	DI2	DI3	DI4	DI5	DI6	DI7	DI8	
	OUTPUT SHIFT REGISTER (OSR)	R	DO1	DO2	DO3	DO4	DO5	DO6	DO7	DO8	
1	STATUS	W	X	X	X	X	X	X	QA1	QA0	
		R	PARITY FAIL	ACTIVE	OSR- EMPTY	ISR- FULL	SERIAL KEY REQ	0	QA1	QA0	
2	PORT CONFIGURATION	R/W	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	
3	MODE CONTROL	R/W	0	0	MC5	MC4	MC3	MC2	MC1	MC0	
4	M1 (PROGRAM MEMORY)	R/W	M17	M16	M15	M14	M13	M12	M11	M10	
5	M2	R/W	M27	M26	M25	M24	M23	M22	M21	M20	
6	M3	R/W	0	0	M35	M34	M33	M32	M31	M30	

TABLE 1. MASTER PORT REGISTERS (READ/WRITE)

SLAVE PORT ADDRESS	REGISTER	READ OR WRITE	CONTENT								LSB
			B7	B6	B5	B4	B3	B2	B1	B0	
0	INPUT SHIFT REGISTER (ISR)	W	DI1	DI2	DI3	DI4	DI5	DI6	DI7	DI8	
	OUTPUT SHIFT REGISTER (OSR)	R	DO1	DO2	DO3	DO4	DO5	DO5	DO7	DO8	
1	STATUS	R	0	ACTIVE	OSR- EMPTY	ISR- FULL	0	0	0	0	

TABLE 2. SLAVE PORT REGISTERS (READ/WRITE)

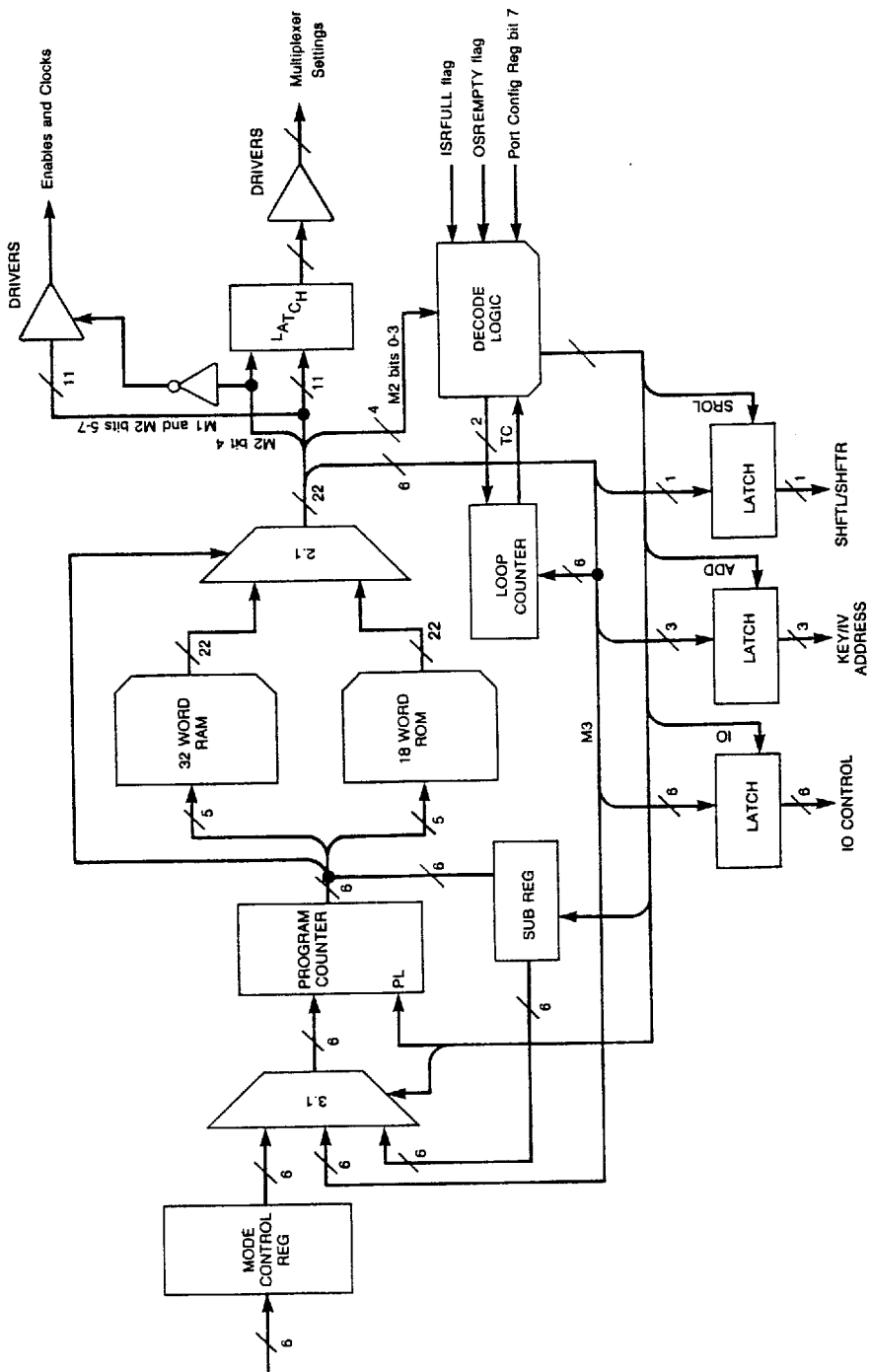


FIGURE 4. PROGRAM SEQUENCER

ADDR	M1	M2	M3	ASSEMBLER MNEMONICS
CODE				
/* DES SUBROUTINE				
/*				
0	c2	1f	0	:00 LDDES CKDES CKKEY
1	42	10	5	:01 CKDES CKKEY LLC 5
2	52	11	2	:02 CKDES SHFT2 CKKEY ILC 02
3	42	10	5	CKDES CKKEY LLC 5
4	52	11	4	:03 CKDES SHFT2 CKKEY ILC 03
5	42	13	0	CKDES CKKEY RET 0
/*				
/* LOAD INITIAL VALUE*				
/*				
6	1	b	3	B6 IO LDMP ACT**
				DES INPUT = ISR OSR INPUT = DESOUT
				IV INPUT = ISR LATCH INPUT = ISR
7	1	1a	0	CLISRF ADD
8	0	15	8	:10 ISRFT? 10
9	0	3c	0	WIV CLEAR
a	0	14	a	:20 GTO 20
/*				
/* PARALLEL LOAD KEY*				
/*				
b	1	b	3	B6 IO LDMP ACT**
				DES INPUT = ISR OSR INPUT = DESOUT
				IV INPUT = ISR LATCH INPUT = ISR
c	1	1a	0	:25 CLISRF ADD
d	0	15	d	:30 ISRFT? 30
e	8	1c	0	WKEY CLEAR
f	0	14	f	:40 GTO 40
/*				
/* SERIAL LOAD KEY*				
/*				
10	1	b	7	B6 IO LDMP SERIAL ACT**
				DES INPUT = ISR OSR INPUT = DESOUT
				IV INPUT = ISR LATCH INPUT = ISR
11	0	14	c	GTO 25

\* These are self contained programs. They may not be called as subroutines from another program.

\*\* B6 is an unnecessary mnemonic in this code.

PROGRAM ROM CODE

TABLE 3

addressed by the status register. The key request flag is then cleared. Odd parity of every eight bits is checked. Each eighth bit input is the parity bit.

At the end of all three of these programs the sequencer goes into an endless loop (wait state) until a new program is executed.

The RAM contains the ciphering program and must be written by the user prior to any ciphering operation. The RAM may hold up to thirty-two instructions, more than enough to program both encrypt and decrypt of any standard DES mode. The user loads the RAM through the eight bit master port. After first writing the RAM address (20H to 3fH) to the mode control register, the user writes three bytes for each 22 bit program instruction. The two most significant bits, in one of the bytes, are not used. The RAM, or the ROM (address 00H to 11H), may be read in a similar manner.

To begin ciphering or the execution of a program located in either RAM or ROM, the user writes the program memory starting address to the mode control register. Two clock cycles later this address is loaded into the program counter (Figure 4) and execution begins. Data flow through the ports and the associated assignments of the master and slave flags are controlled by the port configuration register; see Table 4. Normally this register would be written before executing a ciphering program.

PORT CONFIG	HEX CODE	OUTPUT PIN/FLAG ASSOCIATIONS
MP --> SP	04 OR 84	MFLG1 ~ - ISRFULL SFLG ~ - OSREMPY
MP <-- SP	11 OR 91	MFLG1 ~ - OSREMPY SFLG ~ - ISRFULL
--> MP <--	01 OR 81	MFLG1 ~ - OSREMPY MFLG2 ~ - ISRFULL
MPSD --> SPSPD	28 OR A8	MFLG1 ~ - ISRFULL SFLG ~ - OSREMPY
MPSP <-- SPSPD	62 OR E2	MFLG1 ~ - OSREMPY SFLG ~ - ISRFULL
MP --> SPSPD	08 OR 88	MFLG1 ~ - ISRFULL SFLG ~ - ISRFULL
MP <-- SPSPD	61 OR E1	MFLG ~ - OSREMPY SFLG ~ - ISRFULL

NOTE: THE MOST SIGNIFICANT BIT IN THE HEX CODE FOR THE PORT CONFIGURATION IS AN INPUT FLAG. IT IS TESTED BY THE MICROCODE MNEMONIC LT?. THIS BIT MAY BE USED AS A GENERAL PURPOSE CONDITIONAL JUMP.

**TABLE 4. PORT CONFIGURATION**  
(MASTER PORT ADDRESS = 2)

## MICRO-CODE INSTRUCTION SET

Mnemonics, corresponding to actual signal names, were defined for the program instruction set. Table 5 defines a 22 bit instruction composed of three bytes, M1, M2, and M3.

Bit 4 of M2 controls the interpretation of M1 and the three most significant bits in M2. If bit 4 of M2 is low, the multiplexer select lines are latched. In the program convention used, the presence of a mnemonic, S1 for example, indicates the control line is latched high. Conversely, the absence of a multiplexer mnemonic indicates the control line is latched low. If bit 4 of M2 is high, the specified signal is enabled only for the duration of the instruction period, two clock cycles. An enable and the associated clock signal, e.g., LDDDES and CKDES, must be programmed in the same instruction since none of these signals are latched.

Bits 0 through 3 of M2 are decoded and select one of twelve commands. With the exception of RET and CLEAR, all of these commands use all or some of the bits in M3 as an argument. The three commands SROL, ADD, and IO latch bits of M3 until overwritten, or a subsequent CLEAR command is issued.

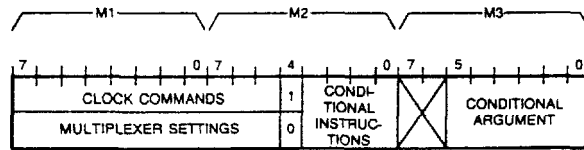
A C language\* assembler was written to facilitate the development of ciphering programs. The output of that assembler is shown in Table 3 for the ROM code. Whenever bit 4 of M2 is set low, the ciphering multiplexers are set-up and the assembler program prints the inputs to the DES (DES INPUT), output shift register (OSR INPUT), initial value register (IV INPUT), and data latch (LATCH INPUT). This is useful in checking that the multiplexer configuration latched is correct. The six instruction DES subroutine may then be explained as follows:

1. The input to the DES initial permutation matrix is clocked into the L and R registers (Figure 2). Simultaneously, the key schedule C and D registers are clocked or shifted. The direction of the shift is dependent on the state of the SHFTR signal. SHFTR is set low to encrypt (left shift) and high to decrypt.
2. The first iteration of the DES is clocked into the L and R registers and the key schedule C and D registers are again

---

\* C is a general purpose programming language designed for and implemented on the UNIX (registered trademark of AT&T Bell Labs) operating system.





INSTRUCTION FORMAT

## CLOCK COMMANDS (M2-bit 4 = 1)

BITS	MNEMONIC	DEFINITION
M1-7	LDOES	Enables the DES enciphering multiplexer to pass the output from MUX1 when high and pass the DES output when low.
-6	CKDES	Clocks the DES L and R registers.
-5	CKL	Clocks the latch register.
-4	SHFT2	Enables the key schedule circuitry to rotate 2 positions when high, and 1 position when low.
-3	WKEY	Write the output from the ISR into the key register currently addressed.
-2	LDKEY	Enables the key schedule circuitry multiplexer to pass the key register output when high and pass the key schedule output when low.
-1	CKKEY	Clocks the key schedule C and D registers.
-0	CLISRF	Clears the ISRFULL flag and allows data to be written into ISR.
M2-7	CLOSRE	Clears the OSREEMPTY flag and allows data to be read from the OSR.
-6	CKOSR	Clocks the output from MUX4 into the OSR.
-5	WIV	Writes the output from MUX13 into the initial value register currently addressed.

## MULTIPLEXER SETTINGS (M2-bit 4 = 0)

BITS	MNEMONIC	DEFINITION	S1	INPUT
M1-7	S1	Selects input line for MUX1.	0 1	0 1
-6	B2	Select input lines for MUX2.	B2	A2
-5	A2		0 0 0 1 1 0 1 1	0 1 2 UNKNOWN
-4	S3	Selects input line for MUX3.	S3	INPUT
-3	S4	Selects input line for MUX4.	S4	INPUT
-2	S5B	Select input lines for MUX5.	S5B	S5A
-1	S5A		0 0 0 1 1 0 1 1	0 1 2 UNKNOWN
-0	B6	Select input lines for MUX6.	B6	A6
M2-7	A6		0 0 0 1 1 0 1 1	0 1 2 UNKNOWN
-6	B13	Select input lines for MUX13.	B13	A13
-5	A13		0 0 0 1 1 0 1 1	0 1 2 3

NOTE: The multiplexer settings are all latched.

TABLE 5. MICRO-CODE INSTRUCTION FORMAT (part 1 of 2)

## CONDITIONAL INSTRUCTIONS

M2-BIT				MNEMONIC	DEFINITION																								
3	2	1	0																										
0	0	0	0	LLC	Loads the loop counter with the least significant nibble in M3. There is only one loop counter.																								
0	0	0	1	ILC	Decrements the loop counter and jumps to the address in M3 if the loop counter is zero.																								
0	0	1	0	SUB	The current program instruction address is incremented and stored before the program jumps to the address specified in M3. Only one level of subroutine call is allowed.																								
0	0	1	1	RET	The program jumps to the address stored when the preceding SUB command was executed.																								
0	1	0	0	GTO	The program jumps to the address in M3.																								
0	1	0	1	ISRFT?	If the ISR is not full, the program jumps to the address specified in M3.																								
0	1	1	0	OSRET?	If the OSR is not empty, the program jumps to the address specified in M3.																								
0	1	1	1	ISRFSRET?	If the ISR is full and the OSR is empty, then the program jumps to the address specified in M3.																								
1	0	0	0	LT?	If bit 7 of the port configuration register is set low, the program jumps to the address specified in M3. This bit may be used to control the order in which the key schedule is invoked.																								
1	0	0	1	SROL	<table><tr><th>BIT</th><th>MNEMONIC</th><th>DEFINITION</th></tr><tr><td>M3-0 = 1</td><td>SHFTR</td><td>Latches a right key schedule rotation.</td></tr><tr><td>M3-0 = 0</td><td>SHFTL</td><td>Latches a left key schedule rotation.</td></tr></table>	BIT	MNEMONIC	DEFINITION	M3-0 = 1	SHFTR	Latches a right key schedule rotation.	M3-0 = 0	SHFTL	Latches a left key schedule rotation.															
					BIT	MNEMONIC	DEFINITION																						
					M3-0 = 1	SHFTR	Latches a right key schedule rotation.																						
M3-0 = 0	SHFTL	Latches a left key schedule rotation.																											
1	0	1	0	ADD	<table><tr><td>M3-0</td><td>INT</td><td rowspan="4">A high latches the internal key/IV address bus. A low latches the external bus (Status Register QA1, QA0)</td></tr><tr><td>M3-1</td><td>ADD0</td></tr><tr><td>M3-2</td><td>ADD1</td></tr><tr><td colspan="2"><table><tr><td></td><td>ADD1</td><td>KEY/IV ADDRESS</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>2</td></tr><tr><td>1</td><td>1</td><td>3</td></tr></table></td></tr></table>	M3-0	INT	A high latches the internal key/IV address bus. A low latches the external bus (Status Register QA1, QA0)	M3-1	ADD0	M3-2	ADD1	<table><tr><td></td><td>ADD1</td><td>KEY/IV ADDRESS</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>2</td></tr><tr><td>1</td><td>1</td><td>3</td></tr></table>			ADD1	KEY/IV ADDRESS	0	0	0	0	1	1	1	0	2	1	1	3
					M3-0	INT	A high latches the internal key/IV address bus. A low latches the external bus (Status Register QA1, QA0)																						
					M3-1	ADD0																							
					M3-2	ADD1																							
<table><tr><td></td><td>ADD1</td><td>KEY/IV ADDRESS</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>2</td></tr><tr><td>1</td><td>1</td><td>3</td></tr></table>			ADD1	KEY/IV ADDRESS	0	0		0	0	1	1	1	0	2	1	1	3												
	ADD1	KEY/IV ADDRESS																											
0	0	0																											
0	1	1																											
1	0	2																											
1	1	3																											
1	0	1	1	IO	<table><tr><td>M3-0</td><td>ACT</td><td>A high latches the ACTIVE flag.</td></tr><tr><td>M3-1</td><td>LDMP</td><td>A high latches the input circuitry to receive data from the master port. Overrides the port configuration setting.</td></tr><tr><td>M3-2</td><td>SERIAL</td><td>A high latches the key circuitry for a serial key input.</td></tr><tr><td>M3-3</td><td>1BIT*</td><td>A high latches the I/O circuitry to write/read a single bit. If the parallel ports are programmed only the most significant bit in the byte is used.</td></tr><tr><td>M3-4</td><td>8BIT*</td><td>A high latches the I/O circuitry to write/read 8 serial bits or 1 parallel byte.</td></tr><tr><td>M3-5</td><td>SISRFSRE</td><td>A high sets both ISRFULL and OSREMP-TY flags active.</td></tr></table>	M3-0	ACT	A high latches the ACTIVE flag.	M3-1	LDMP	A high latches the input circuitry to receive data from the master port. Overrides the port configuration setting.	M3-2	SERIAL	A high latches the key circuitry for a serial key input.	M3-3	1BIT*	A high latches the I/O circuitry to write/read a single bit. If the parallel ports are programmed only the most significant bit in the byte is used.	M3-4	8BIT*	A high latches the I/O circuitry to write/read 8 serial bits or 1 parallel byte.	M3-5	SISRFSRE	A high sets both ISRFULL and OSREMP-TY flags active.						
					M3-0	ACT	A high latches the ACTIVE flag.																						
					M3-1	LDMP	A high latches the input circuitry to receive data from the master port. Overrides the port configuration setting.																						
					M3-2	SERIAL	A high latches the key circuitry for a serial key input.																						
					M3-3	1BIT*	A high latches the I/O circuitry to write/read a single bit. If the parallel ports are programmed only the most significant bit in the byte is used.																						
					M3-4	8BIT*	A high latches the I/O circuitry to write/read 8 serial bits or 1 parallel byte.																						
M3-5	SISRFSRE	A high sets both ISRFULL and OSREMP-TY flags active.																											
1	1	0	0	CLEAR	A high sets all bits in the latches controlled by SROL, ADD, and IO low.																								

\*NOTE: If both of these bits are low, the I/O circuitry is set to write/read 64 serial bits or 8 parallel bytes. The condition with both bits being high is undefined.

TABLE 5. MICRO-CODE INSTRUCTION FORMAT (part 2 of 2)

shifted one position. A five is loaded into the loop counter.

3. This statement is executed six times as the next six DES iterations are clocked into the L and R registers. Simultaneously, the key is shifted two positions six times.
4. The eighth iteration of the DES is clocked into the L and R registers and the key schedule C and D registers are again shifted one position. A five is loaded into the loop counter.
5. This statement is executed six times as the next six DES iterations are clocked into the L and R registers. Simultaneously, the key is shifted two positions six times.
6. The fifteenth iteration of the DES is clocked into the L and R registers and the key schedule C and D registers are again shifted one position.

At this point the output of the DES enciphering circuitry, the inverse initial permutation matrix (IP-1), will have the sixteenth DES iteration or the output block.

A sample of the RAM micro-code for the standard ECB and CBC operating modes is given in Table 6. The code for the remaining standard operating modes is available and documented.

## APPLICATIONS

The following applications illustrate the unique capabilities of the DEP. In order to perform similar operations with available integrated DES devices, considerable processor overhead or multiple devices might be required.

### TWO WAY ENCRYPTION APPLICATION

The first application describes a two way encryption system using separate receive and transmit keys. A drop-in box between a terminal (or computer) and a modem was built. Clearly, this system requires a character oriented protocol. The eight bit cipher feedback mode was used. In a typical terminal to computer connection, the number of characters transmitted and received are unequal. The ciphering operation desired is shown in Figure 5.

To transmit an encrypted character the number in initial value

ADDR	M1	M2	M3	CODE	ASSEMBLER MNEMONICS
/* ECB ENCRYPT OR DECRYPT					
/*					
20	1	c	0	B6	CLEAR
				DES INPUT =	ISR OSR INPUT = DESOUT
				IV INPUT =	ISR LATCH INPUT = ISR
21	7	18	23	LDKEY CKKEY	CLISRF LT? 100
22	2	19	1	CKKEY SROL	SHFTR
23	0	15	23	:	100 ISRFT? 100
24	c3	12	1	CLISRF LDDDES CKDES CKKEY	SUB 01
25	0	17	28	ISRFOSRET?	120
26	0	16	26	:	110 OSRET? 110
27	0	d4	23	CLOSRE CKOSR	GTO 100
28	c3	d2	1	:120 CLISRF CLOSRE CKOSR LDDDES CKDES CKKEY	SUB 01
29	0	17	28	:	130 ISRFOSRET? 120
2a	0	14	26	GTO	110
/*					
/* CBC ENCRYPT					
/*					
2b	3	c	0	S5A B6	CLEAR
				DES INPUT =	ISR^IV OSR INPUT = DESOUT
				IV INPUT =	ISR LATCH INPUT = ISR^IV
2c	7	18	2e	LDKEY CKKEY	CLISRF LT? 200
2d	2	19	1	CKKEY SROL	SHFTR
2e	0	15	2e	:	200 ISRFT? 200
2f	c3	12	1	CLISRF LDDDES CKDES CKKEY	SUB 01
30	13	4	29	:	210 S3 S5A B6 GTO 130
				DES INPUT =	ISR^DESOUT OSR INPUT = DESOUT
				IV INPUT =	ISR LATCH INPUT = ISR^DESOUT
/*					
/* CBC DECRYPT					
/*					
31	7	1c	0	LDKEY CKKEY	CLISRF CLEAR
32	59	48	34	B2 S3 S4 B6 B13	LT? 250
				DES INPUT =	ISR OSR INPUT = IV^DESOUT
				IV INPUT =	Qn LATCH INPUT = ISR
33	2	19	1	CKKEY SROL	SHFTR
34	0	15	34	:	250 ISRFT? 250
35	e3	12	1	CLISRF CKL LDDDES CKDES CKKEY	SUB 01
36	0	17	39	ISRFOSRET?	230
37	0	16	37	:	220 OSRET? 220
38	0	f4	34	CLOSRE CKOSR WIV	GTO 250
39	e3	f2	1	:230 CLISRF CKL WIV CLOSRE CKOSR LDDDES CKDES CKKEY	SUB 01
3a	0	17	39	ISRFOSRET?	230
3b	0	14	37	GTO	220

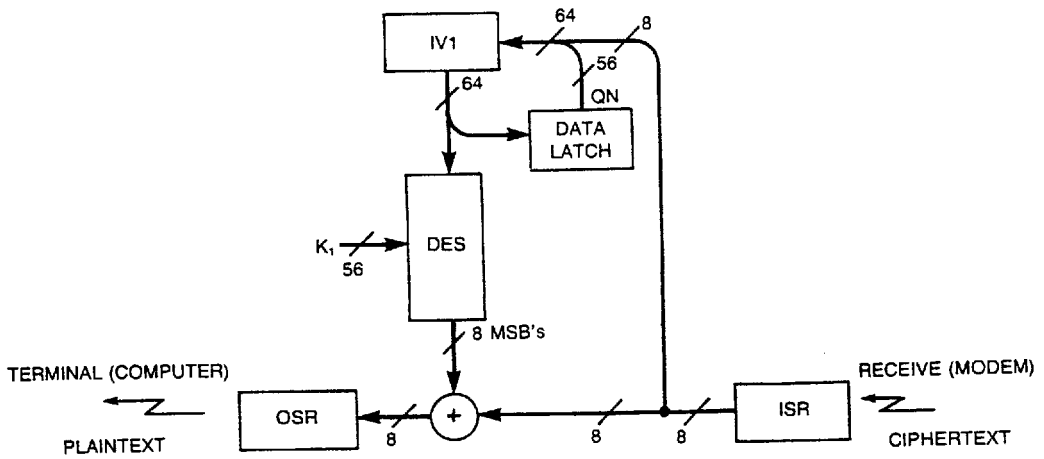
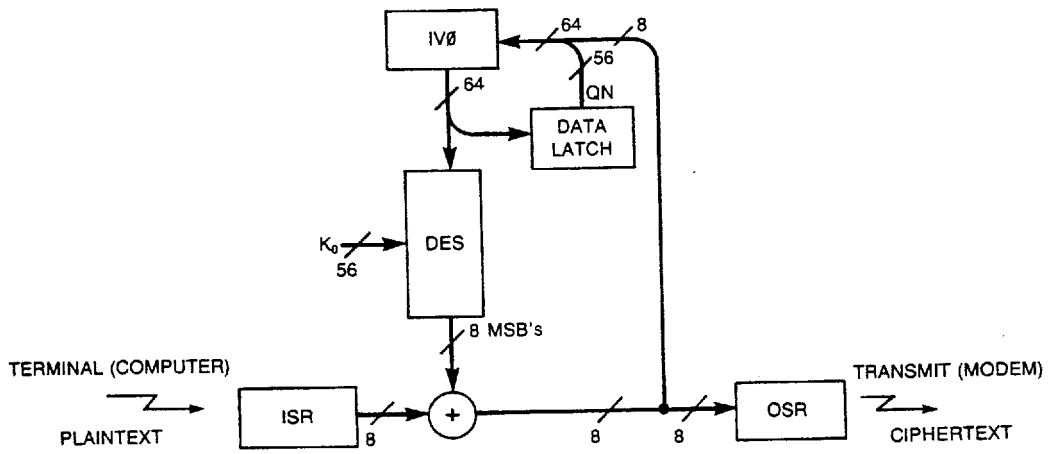
## ASSEMBLER OUTPUT CODE FORMAT

Example: 31 7 1c 0  
 four hex bytes  
 31 Memory address to be loaded into the MODE CONTROL REGISTER  
 07 M1 byte  
 1c M2 byte  
 00 M3 byte

NOTE: User programs must be written between hex addresses 20 and 3f hex, inclusive.

RAM CODE FOR ECB AND CBC OPERATING MODES

TABLE 6



TWO WAY ENCRYPTION

FIGURE 5.

register 0 is encrypted using key register 0. As this number is being clocked into the DES enciphering hardware, it is also clocked into the data latch. When a plaintext character is input, it is added modulo 2 to the eight most significant bits in the DES output block,  $\text{DESOUT} \wedge \text{ISR}$ . (The symbol  $\wedge$  is used to define the "exclusive-or" operator.) This byte of ciphertext output is clocked into the output shift register for transmission by the modem. It is also clocked into initial value register 0 as the least significant byte. The seven other bytes are simply the previous initial value shifted one byte to the left. The most significant byte of the previous initial value is discarded.

The receive operation is nearly identical. The number in initial value register 1 is encrypted using key register 1. As this number is being clocked into the DES enciphering hardware it is also clocked into the data latch. When a ciphertext character is input, it is added modulo 2 to the eight most significant bits in the DES output block. This byte of plaintext output is clocked into the output shift register for reception by the local terminal (computer). The ciphertext in the input shift register is clocked into initial value register 1 as the least significant byte. The seven other bytes are simply the previous initial value shifted one byte to the left. The most significant byte of the previous initial value is discarded.

There are two differences, then, between transmit and receive. One difference is the feedback to the initial value register. During transmit,  $\text{ISR} \wedge \text{DESOUT}$  is fed back. During receive, only ISR is fed back. The second difference is that key/initial value register pair 0 is used in transmit and pair 1 is used in receive.

Code for this ciphering mode is shown in Table 7. The statement " $\text{DES INPUT} = \text{Qn} \ll 8 \mid \mid \text{ISR} \wedge \text{DESOUT}$ ", taken from Table 7, should be read as follows: the input to the DES enciphering block equals the data latch output (Qn) shifted eight bits to the left and concatenated with the eight most significant bits in the "exclusive-or" sum of the input shift register and the DES enciphering block output. After loading the RAM program memory with the hex data in Table 7, the program start address (2bH) is written to the mode control register and execution begins. The program will remain in a loop (2cH to 2fH) until the input shift register is filled. Depending on the most significant bit in the port configuration register, the DEP will either encrypt (transmit) using key/initial value register pair 0 or decrypt

```

CODE
ADDR M1 M2 M3      ASSEMBLER MNEMONICS

/* ECB ENCRYPT OR DECRYPT
/*
20 1 c 0      B6 CLEAR
      DES INPUT = ISR      OSR INPUT = DESOUT
      IV INPUT = ISR      LATCH INPUT = ISR
      LDKEY CKKEY CLISRF LT? 100
      CKKEY SROL SHFTR
      :100 ISRF? 100
      CLISRF LDDDES CKDES CKKEY SUB 01
      ISRFOSRET? 120
      :110 OSRET? 110
      CLOSRE CKOSR GTO 100
      :120 CLISRF CLOSRE CKOSR LDDDES CKDES CKKEY SUB 01
      :130 ISRFOSRET? 120
      GTO 110
/* 8 BIT CFB ENCRYPT INTERNAL KEY-IV 0
/* OR DECRYPT INTERNAL KEY-IV 1
/* MIN OF 26 INST IN DECRYPT
/*
2b 1 1b 10      CLISRF IO 8BIT
2c 1d 2a 1      :101 S3 S4 S5B B6 A13 ADD INT
      DES INPUT = IV      OSR INPUT = ISR^DESOUT
      IV INPUT = IV      LATCH INPUT = IV
      LT? 102
      ADD INT ADD0
      :102 ISRF? 101
      CKL LDKEY CKKEY SUB 00
      :103 OSRET? 103
      CKOSR CLOSRE
      S3 S4 S5A A6 A13 LT? 104
      DES INPUT = Qn<<8 || ISR^DESOUT      OSR INPUT = ISR^DESOUT
      IV INPUT = Qn<<8 || ISR^DESOUT      LATCH INPUT = Qn<<8 || ISR^DESOUT
      S3 S4 A6 A13
      DES INPUT = Qn<<8 || ISR      OSR INPUT = ISR^DESOUT
      IV INPUT = Qn<<8 || ISR      LATCH INPUT = Qn<<8 || ISR
      :104 CLISRF WIV GTO 101

```

RAM PROGRAM CODE FOR THE TWO WAY ENCRYPTION SYSTEM

TABLE 7

(receive) using key/initial value register pair 1. The mnemonic LT? is used to test the most significant port bit. A low is used for transmit (jump condition) and a high for receive (next instruction). The only timing requirement on the input to the DEP, when changing from transmit to receive, is that the data byte written be delayed from the port register write by three DEP program instructions. This guarantees the LT? instruction (2dH) will be executed after the port register write and before data ciphering. With a 4 Mhz DEP clock, this is 1.5 microseconds. After the data byte is written, the DEP program sequencer will detect an input shift register full condition and cipher the data. If the previous output data has been read, the new cipher byte will be written to the output shift register; the next initial value will be stored; and the sequencer will again cycle waiting for the input shift register to be filled. If the previous output data has not been read, the sequencer will wait (31H) until the output shift register is emptied. It will take at most twenty-four instructions from the time an input byte is written until the cipher text is available to be read. For a 4 Mhz clock, this is twelve microseconds.

In order for two stations to communicate properly, if k0 and k1 are input to key registers 0 and 1 (respectively) of a DEP device at station one, then k0 and k1 must be input to key registers 1 and 0 (respectively) of the DEP device at station two. The two stations need not have the same initial value, since a station will synchronize after eight characters have been received. This is a property of the eight bit CFB mode. Therefore, to begin a session the two stations only have to establish session keys. The protocol shown in Figure 6 was used to exchange session keys. This protocol does not require either station to be a master or slave; both stations perform exactly the same operations. A master key is input to key register 2. A random number loaded into key register 0 is encrypted in the ECB mode under the master key. This ciphertext is then transmitted. The received ciphertext is decrypted and loaded into key register 1. After these three operations, the session key exchange is complete and two way communications may begin.

Before this system could become a viable encryption product some additional work should be done. The error rates over the public telephone network combined with the eight byte error extension property of the CFB mode results in an unacceptable error rate in the



decoded plaintext. The OFB mode does not have the error extension property associated with CFB. A single transmission bit error results in a single plaintext bit error, however, telephone line noise frequently generates characters never legitimately transmitted. In the OFB mode or in any other key stream mode this would result in a loss of synchronization. This condition would have to be detected and initial values re-established, very messy. Since ASCII is a seven bit code, programming the DEP for 7 bit CFB, appending parity, buffering several bytes, and retransmitting bytes in the event of a parity failure would substantially reduce the error rate. In the current system a single ECB encryption of the session key is performed. It is suggested that a double or even a triple encryption of the session key be done since master keys would probably be changed infrequently. Some check should be made for transmission errors in exchanging session keys. If the wrong session key is used, nothing will be decoded in one direction.

#### TRIPLE ENCRYPTION APPLICATION

In the second application, the DEP is programmed for a triple encryption. Such a program might be used to increase security in applications involving very sensitive or valuable data. The use of multiple keys to encrypt the data effectively increases the key space an intruder must search to decode the ciphertext. Three separate cipher block chaining (CBC) operations are performed on a single DES input block. Three different keys and initial values, register pairs 0, 1 and 2, are used for the ciphering. The data latch is needed in the decrypt operation to hold intermediate products. Figure 7 shows the ciphering mode and Table 8 lists the code.

The first instruction in the CBC encrypt code is to clear the input shift register so the user may begin loading data. The multiplexers are then setup with the input to the DES enciphering circuitry equal to the "exclusive-or" sum of the input shift register and initial value register. In this same instruction, the address of key/initial value register pair 0 is latched. Nothing further happens until the input shift register is filled. When that occurs: the first ciphering operation is performed; the input shift register is cleared so the second block of data may be entered; the DES output is written to initial value register 0; and register pair 1 is addressed. Next,

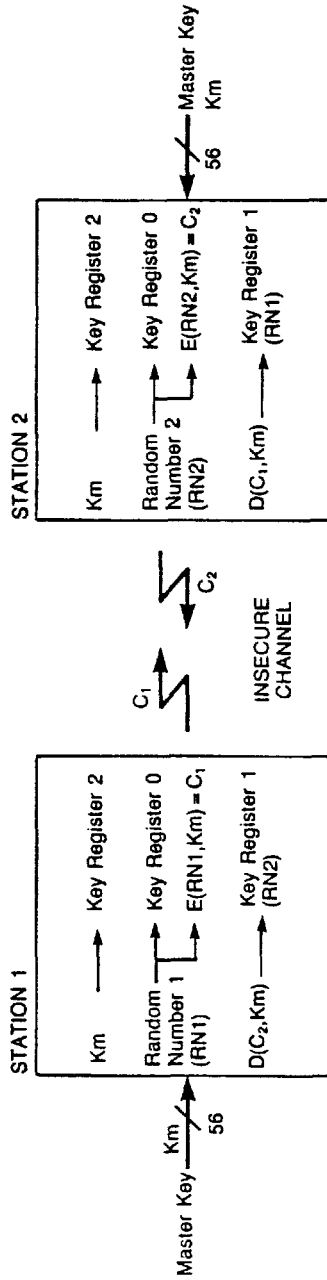
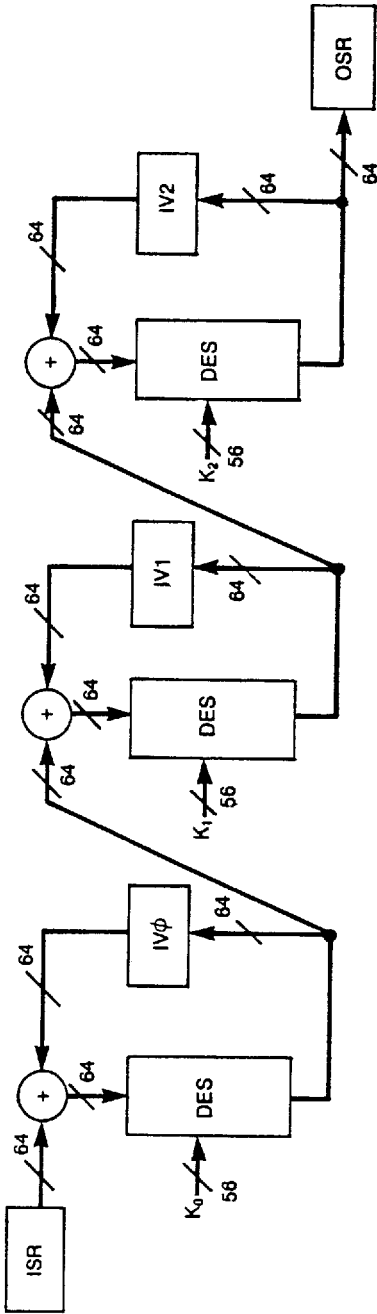


FIGURE 6. SYMMETRICAL PROTOCOL FOR SESSION KEY EXCHANGE

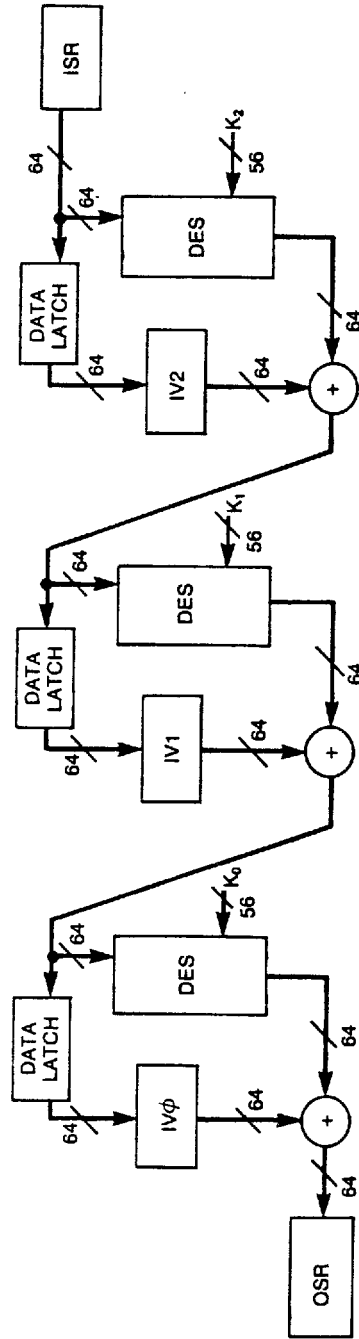
the multiplexers are reset so the input to the DES comes from the "exclusive-or" sum of the current DES output and initial value register 1. The second key (key register 1) and the new DES block are clocked into the DES circuitry. The key is shifted one position to the left and the DES subroutine is called. This completes the second ciphering iteration. The DES output is written to initial value register 1, and register pair 2 is addressed. The last ciphering operation is performed. The sequencer waits until the output shift register flag is set before clocking that register, clearing the output shift register empty flag, and jumping back to the second program instruction (21H).

The decrypt is similar to the encrypt operation with certain important exceptions. The keys and initial value register pairs must be invoked in reverse order. Similarly, the DES key schedule must be reversed. Hence, the instruction CKKEY SROL SHFTR (30H, 36H and 3aH) is required. This sets the key schedule circuitry for a right instead of a left shift. A third difference is that the data latch holds the new initial value while the current one is being used. Consequently, the decryption code requires three more statements than the encryption code. In decryption, sixty-one program instructions are executed, provided there is no waiting for the input shift register to be loaded or the output shift register to be emptied. With a 4 Mhz clock, 32.8K ciphering operations per second could be performed. This triple encryption takes 3.6 times as long as a single encryption (seventeen program instructions), so the additional overhead is only 20%.

There are many ways to implement cascaded ciphers and to feed back data blocks. The one just described suffers from error propagation. A single error in the input block to the decryption chain results in a 50% error rate in the current and the next two output blocks as well as a single bit error in the fourth block. At CRYPTO 84 Adi Shamir suggested two possible configurations for cascaded ciphers with no additional error propagation and no "meet in the middle" known plaintext attacks. These are shown in Figure 8. Under a known plaintext attack, if the initial value is kept secret, the DES input remains unknown for these configurations. The DEP may be easily programmed for either of these modes.



CBC ENCRYPT — 3 KEYS AND INITIAL VALUES



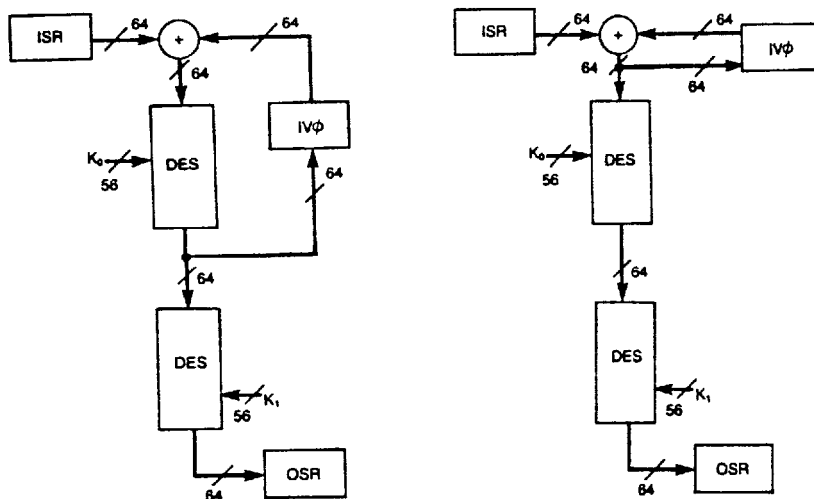
CBC DECRYPT — 3 KEYS AND INITIAL VALUES

FIGURE 7.

ADDR	M1	M2	M3	ASSEMBLER MNEMONICS
CODE				
/* CBC ENCRYPT 3 KEYS				
/*				
20	1	1c	0	CLISRF CLEAR
21	3	6a	1	:50 S5A B6 B13 A13 ADD INT
				DES INPUT = ISR^IV OSR INPUT = DESOUT
				IV INPUT = DESOUT LATCH INPUT = ISR^IV
22	0	15	22	:60 ISRFT? 60
23	7	12	0	CLISRF LDKEY CKKEY SUB 00
24	0	3a	3	WIV ADD INT ADD0
25	53	6f	0	S3 B2 S5A B6 B13 A13
				DES INPUT = IV^DESOUT OSR INPUT = DESOUT
				IV INPUT = DESOUT LATCH INPUT = IV^DESOUT
26	c6	1f	0	LDKEY CKKEY LDDDES CKDES
27	2	12	1	CKKEY SUB 01
28	0	3a	5	WIV ADD INT ADD1
29	c6	1f	0	LDKEY CKKEY LDDDES CKDES
2a	2	12	1	CKKEY SUB 01
2b	0	16	2b	:70 OSRET? 70
2c	0	f4	21	WIV CLOSRE CKOSR GTO 50
/*				
/* CBC DECRYPT 3 KEYS				
/*				
2d	1	1c	0	CLISRF CLEAR
2e	59	4a	5	:160 B2 S3 S4 B6 B13 ADD INT ADD1
				DES INPUT = ISR OSR INPUT = IV^DESOUT
				IV INPUT = Qn LATCH INPUT = ISR
2f	6	19	0	LDKEY CKKEY SROL SHFTL
30	2	19	1	CKKEY SROL SHFTR
31	0	15	31	:170 ISRFT? 170
32	e3	12	1	CLISRF CKL LDDDES CKDES CKKEY SUB 01
33	5b	4f	0	B2 S3 S4 S5A B6 B13
				DES INPUT = IV^DESOUT OSR INPUT = IV^DESOUT
				IV INPUT = Qn LATCH INPUT = IV^DESOUT
34	e0	3a	3	CKL LDDDES CKDES WIV ADD INT ADD0
35	6	19	0	LDKEY CKKEY SROL SHFTL
36	2	19	1	CKKEY SROL SHFTR
37	2	12	1	CKKEY SUB 01
38	e0	3a	1	CKL LDDDES CKDES WIV ADD INT
39	6	19	0	LDKEY CKKEY SROL SHFTL
3a	2	19	1	CKKEY SROL SHFTR
3b	2	12	1	CKKEY SUB 01
3c	0	16	3c	:180 OSRET? 180
3d	0	f4	2e	WIV CLOSRE CKOSR GTO 160

RAM PROGRAM CODE FOR TRIPLE ENCRYPTION

TABLE 8



TWO CASCADE CIPHERS (SUGGESTED BY ADI SHAMIR)

FIGURE 8

## CONCLUSIONS

A user programmed Digital Encryption Processor based on the National Bureau of Standards DES algorithm has been described. The DEP has been certified by the NBS as complying with the DES. All four of the NBS defined operating modes may be programmed. Multiple (cascaded) or multiplexed ciphering operations may be programmed, eliminating the need for more than one encryption device in some

applications. The internal program sequencer allows the user to tailor the ciphering function for the specific system application. These features place the DEP beyond existing commercial devices. In order to extend the life of the DES, we would like to see more secure modes developed and analyzed. The DEP may be programmed to perform cascaded ciphering using all four key registers. The data throughput rate of 0.59 megabytes per second, for the standard modes under worst case conditions, is comparable with the fastest commercial part now available. For some of the unique modes, the data rate will be much faster since there is no host processor overhead.

The proliferation of smart terminals and computers is leading to distributed networks with access to large data bases. These networks, along with the booming cable television market and satellite communications networks, are prime candidates for low cost secure encryption.

#### REFERENCES

- [1] Federal Information Processing Standards Publication 46, "Data Encryption Standard," January 15, 1977, published by the National Bureau of Standards.
- [2] Federal Information Processing Standards Publication 81, "DES Modes of Operation," December 2, 1980, published by the National Bureau of Standards.
- [3] W. Diffie and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, June 1977.
- [4] S. Even and O. Goldreich, "On the Power of Cascade Ciphers," Advances in Cryptology, Proceedings of Crypto 83.
- [5] Whitfield Diffie and Martin E. Hellman, "Privacy and Authentication to Cryptography", Proceeding of the IEEE, Vol. 67, No. 3, March 1979.
- [6] Alan G. Konheim, "Cryptography: A Primer", John Wiley and Sons, INC., 1981, chapter 6.

# Efficient hardware and software implementations for the DES

Marc Davio<sup>1,3</sup>, Yvo Desmedt<sup>2</sup>, Jo Goubert<sup>2</sup>, Frank Hoornaert<sup>2</sup> and  
Jean-Jacques Quisquater<sup>1</sup>

<sup>1</sup> Philips Research Laboratory, Avenue Van Becelaere, 2,  
B-1170 Brussels, Belgium;

<sup>2</sup> Katholieke Universiteit Leuven, Laboratorium ESAT,  
Kardinaal Mercierlaan, 94, B-3030 Heverlee, Belgium;

<sup>3</sup> Université Catholique de Louvain, Batiment Maxwell,  
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium.

## Abstract

**Importance of DES:** NBS, ANSI and ISO (in study) have DES as standards.

The available devices or programs have some tedious properties for an extensive use:

- hardware is expensive or slow, and limited,
- software is slow.

We describe methods for obtaining efficient hardware and software implementations for the DES, i.e.:

### Hardware

- Cheap and fast hardware,
- all standard modes,
- available for IC library;

### Software

- Fast i.e. 150 kbit/s (VAX 11/780 without accelerator),
- possibility of using small microprocessors (i.e. small programs with relative high speeds).

These efficient designs are obtained using, e.g., tables which are distinct from the tables described by the NBS norm. *This leads to new problems for testing and for certification.*

### Tools

#### General

- DES paper presented at CRYPTO-83,
- further simplifications,
- analysis of modes;

#### Hardware

- Taking the routing problems in consideration;



## Software

- Precomputations of some tables,
- using *effectively* the size of words of the processor (8, 16, 32, 48) and the available operations.

## Common techniques

### CRYPTO-83 paper

- Analytical properties:  $IP$ ,  $E$  and  $PC_1$ ,
- equivalent representations: iterative DES, modification of the table  $P$ .

### Feedback modes

#### Idea

- The idea is to put  $IP$  as close as possible at the input of the feedback and  $IP^{-1}$  as close as possible to the output of the feedback. This simplifies the routing and the clock circuits in a hardware implementation. For the hardware and for the software, if possible, one can then perform  $IP$ ,  $IP^{-1}$  and DES' in parallel, where DES' is a  $IP$ -free DES. Another reason is related to the security of the implementation (key confinement).

### Key generation

- Precomputation versus parallel computation.

$P \cdot E$ : Analytical expression.

*Remark.* The key remains constant in the four DES modes.

## Software

Good software designs for the DES are obtained using algorithm transformations; for instance, function composition, good match with primitives of the used processor, pre-computations. Some time-memory tradeoffs are necessary in order to avoid too expensive tables. The term "too expensive" is relative to a given processor: a small microprocessor has only 8 registers of one byte, 100 bytes of internal RAM and 1000 bytes of program while a big computer is composed of about 10 megabytes of data and program (not using the virtual memory which gives bad performances in very repetitive tasks).

Use of the CRYPTO-83 paper:

- Analytical properties:  $IP$ ,  $E$ ,  $PC_1$ ,
- equivalent representations,
- idea of  $P \cdot E$ ,
- iterative DES,
- modification of the table  $P$ .

Special technique exists for  $E$ . Another technique is the *precomputation of the key scheduling*. This precomputation requires 96 bytes of RAM for storing the 16 intermediate keys. For some microprocessors, this value is prohibitive: other techniques with precomputations exist with only 16 bytes of RAM but using more complicate procedures.

The use of a two-stage iterative DES model simplifies the program, using the fact that DES is sequential in nature.

The precomputations of tables are useful for obtaining fast software. For instance, each  $S$ -box realizes four functions of six variables, i.e. requires 256 bits of memory (total: 256 bytes for 8  $S$ -boxes). If we realize these  $S$ -boxes as four groups of two  $S$ -boxes, we now need 32 kbytes, but the number of accesses to the  $S$ -boxes has been halved.

Other technique is to combine the  $S$ -boxes with the permutation  $P$ . This technique demands 2096 bytes of memory.

The 48 bit model (see CRYPTO-83 paper) is very useful on computers with words of 48 bits.

A software implementation of the DES on a VAX 11/780 has been made. The expected speed of about 150 kbits has been obtained. Other implementations are studied. A complete paper will appear in the near future.

## Hardware

See the relevant paper in these proceedings.

# Efficient hardware implementation of the DES

Frank Hoornaert, Jo Goubert, and Yvo Desmedt

Katholieke Universiteit Leuven,  
Laboratorium ESAT,  
Kardinaal Mercierlaan, 94,  
B-3030 Heverlee, Belgium.

**Abstract.** Several improvements to realize implementations for DES are discussed. One proves that the initial permutation and the inverse initial permutation can be located at the input, respectively the output of each mode in DES. A realistic design for an exhaustive key search machine is presented.

## 1. Introduction

In [14] the reader find that hardware which is at the same time cheap and fast does not exist [1]. In this paper we propose mainly one efficient chip design for the DES. Nevertheless, depending on the need of the user, different versions are possible. *The proposed version is designed for general purpose.* In section 10 however *a version for exhaustive search of the keys* is illustrated. Important is that all versions can use the same techniques.

The reader not familiar with the DES finds the NBS description of the DES in the literature [9].

### 1.1. Problems and used techniques

By designing the chip one has to solve several technical problems as:

1. the complexity of the routing
2. the minimization of the needed chip area
3. the maximization of the desired speed
4. the limitation of pin connections.

The main problem in a DES chip is the routing. This can be illustrated by looking at figure 1 which shows a straightforward realization of the DES-algorithm [4]. There we see that the width of the interconnections (e.g. 64, 48 ....) will result in an enormous loss of area and speed.

We can distinct 5 important techniques (*see list*) to solve previous problems.

1. a serial/parallel realization with shiftregisters of the permutations and of memory and internal transport in order to reduce the number of interconnections on chip (sections 3 and 4)
2. the use of equivalent representations of the DES in order to optimize speed (section 5)
3. pipelining of the different datablocks in order to maximize the activity on the chip (section 6)
4. rearrangement of the functional elements in order to shorten the length of interconnections (section 7)
5. a modular controller with microprogramming in order to ensure the flexibility of the design (section 8)

Remark that as consequence of the needed coherence in the hardware solution, a lot of simplifications proposed in previous papers (e.g. [3]) couldn't be used.

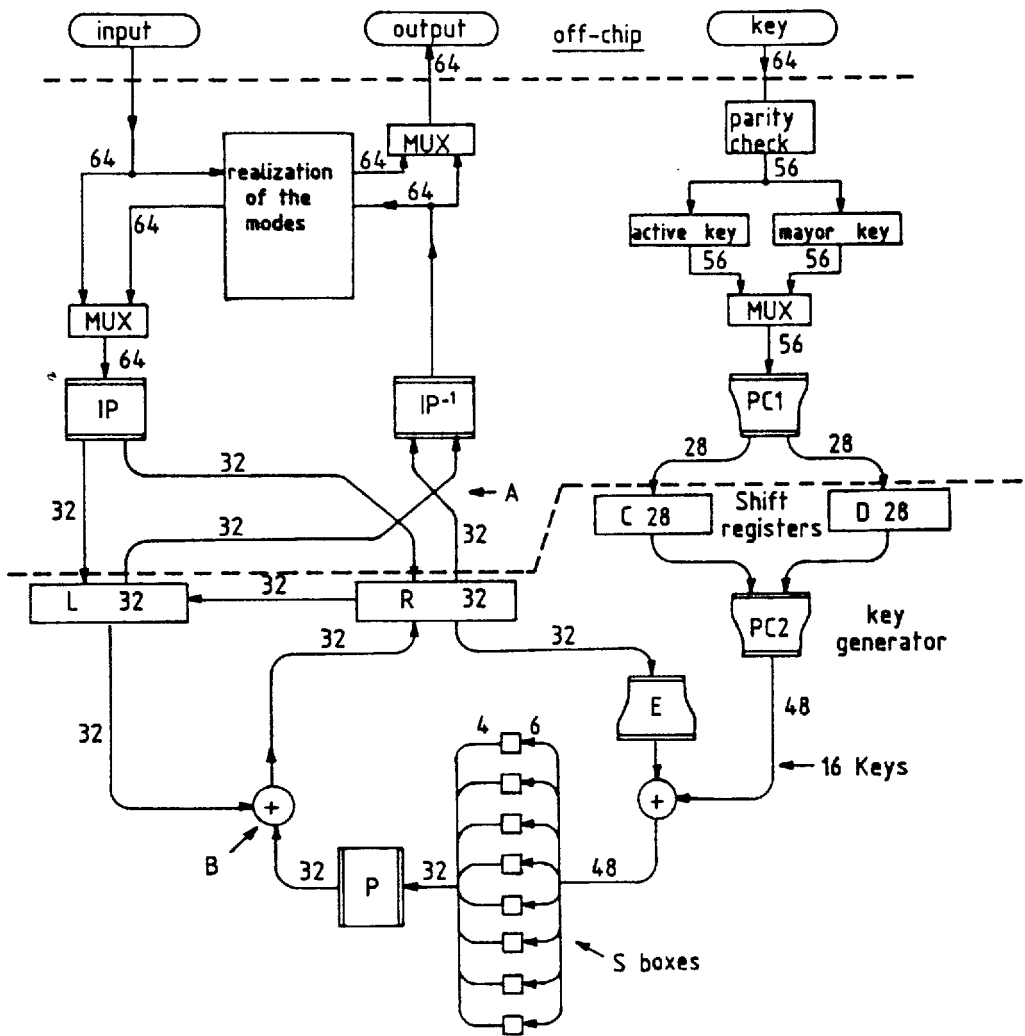
In the next sections the routing problem for the transport part of the chip is solved by using a serial-parallel structure. At the same time, we reduce the area needed for  $IP$ ,  $IP^{-1}$  and  $PC_1$  to about 1/20 of the area compared with a full parallel realization. *Important is that all this solutions do not slow down the data rate.*

The routing problem for the part which carries out the 16 iterations (incl. the subkey generation), is solved by rearranging the different elements. So we shorten the length of the interconnections. This will be explained in extension in section 7.

## 1.2. survey of the chip

We divide the chip in 2 important parts. The first called *transport part*, supports the data transport with the environment and also the internal transport between different memories (incl. the permutations  $IP$ ,  $IP^{-1}$  and  $PC_1$ ). The second called *iteration hardware*, calculates the 16 iterations of the DES-algorithm without  $IP$  and  $IP^{-1}$  (*from now on we call these 16 iterations DES\**) incl. the generation of the subkeys.

We will first explain the basic idea used to simplify the realization of the modes. Later on we will explain the other techniques used.



the upper part of the figure is the transport part  
the lowest part shows the iteration hardware

On the first sight, the DES algorithm suffers from an enormous routingproblem. Consequently the main goal of our design was to solve this problem by serialization and rearrangement without slowing down datarate.

Figure 1: the routing problem

## 2. Equivalent representation for the modes: An introduction

In the Fips publication [8], four modes of operation for the DES are defined. These modes specify different ways to encrypt and decrypt data. We show that you can reorder these modes so that  $IP$  and  $IP^{-1}$  appear at an other location in the algorithm. We'll explain first the general idea and the motivation of this transformation. Next we'll apply this idea in the case of the 8 byte modes and the 1 byte modes.

Mostly the execution of permutations in hard- or software slows down the performances. The idea is to put  $IP$  respectively  $IP^{-1}$  as close as possible to the input respectively the output of the mode. So you don't have to carry out  $IP$  and  $IP^{-1}$  on the data in the feedbackloop. To move the permutations, we'll use the following properties. A permutation followed by a selection, can always be transformed into a selection\* followed by a permutation\* (figure 2). A similar remark is true for an injection followed up by a permutation. The elementary transformations as explained at crypto 83 will also be used [3].

### 2.1. 8 BYTES MODES

In the case of the ECB mode it's trivial that  $IP$  is located at the input and  $IP^{-1}$  at the output.

In the CFB mode we can write on location A (figure 3)  $IP \cdot IP^{-1}$  and propagate them over the exors, using the elementary transformations of Crypto-83 [3] page 182, we obtain then the desired result.

A similar result is similar to find for the CBC and OFB modes.

### 2.2. 1 BYTE MODES

In CFB mode, a new input for the DES is formed out of the previous input for the DES and the actual output ciphertext. The 56 most significant bits of the new input for the DES come from the old input shifted 8 times to the left. This can be represented (figure 4) as a selection of 56 bits out of 64 bits ( $S_1$ ) together with an injection of 56 bits into 64 bits ( $I_1$ ). The 8 least significant bits of the new input for the DES are the 8 bits output ciphertext. This can be represented by an injection  $I_2$  of those 8 bits into 64 bits. To form the output ciphertext the 8 most significant bits of the DES output are selected by selection  $S_2$ , and exored with the plaintext. The content of these selections and injections is showed in table 1 and 2. The selections and injections are similar represented as the permutations of the DES in the NBS norm [9]. Now by applying some transformations one is able to obtain the desired result.

Let us therefore put  $IP \cdot IP^{-1}$  at location B in figure 4. Using the property of figure 2 we obtain figure 5, where the effect of  $Q_a(S_a(64bits))$  is the same as  $S_2(IP^{-1}(64bits))$ . In figure 5 we put  $Q_a^{-1} \cdot Q_a$  at location C. By moving  $Q_a$ ,  $IP$  and  $IP^{-1}$  (using Crypto 83) we obtain figure 6. Using the first property of figure 4, this figure is transformed into figure 7, where  $Q_b(S_b(64bits))$  is the same as  $S_1(IP(64bits))$  and similar for  $I_c(Q_c)$  and

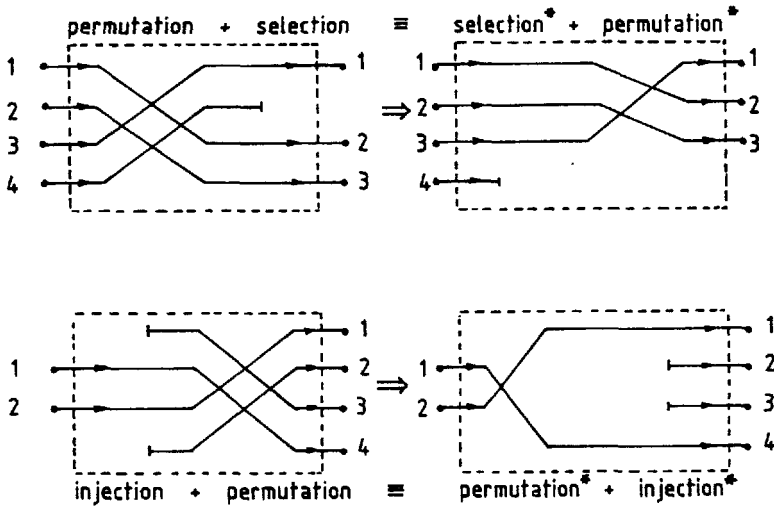


Figure 2: properties used in the transformation of the modes

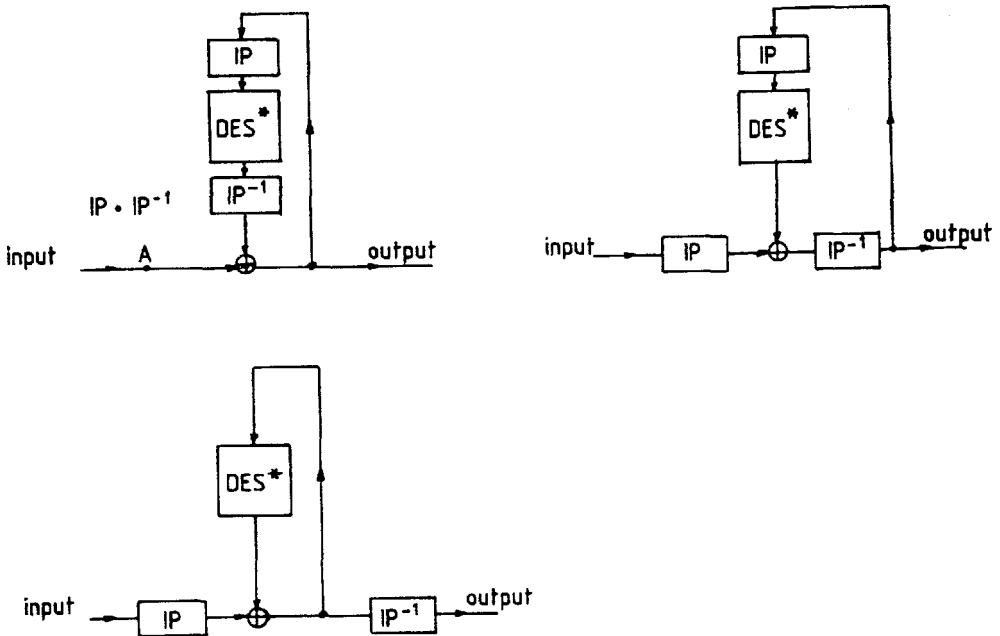
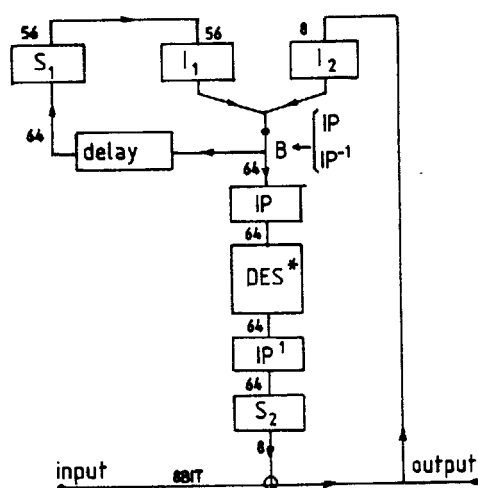


Figure 3: CFB 8 byte mode



$S$  = a selection  
 $I$  = an injection

Figure 4: representation of the 1 byte CFB mode with selections and injections

$S_1 = [$	9	10	11	12	13	14	15	16		$S_2 = [$	1	2	3	4	5	6	7	8	$]$
	17	18	19	20	21	22	23	24											
	25	26	27	28	29	30	31	32											
	33	34	35	36	37	38	39	40											
	41	42	43	44	45	46	47	48											
	49	50	51	52	53	54	55	56											
	57	58	59	60	61	62	63	64	$]$										

Table 1: the selections  $S_1$  (64  $\rightarrow$  56) and  $S_2$  (64  $\rightarrow$  8)

$I_1 = [$	1	2	3	4	5	6	7	8		$I_2 = [$	x	x	x	x	x	x	x	x	
	9	10	11	12	13	14	15	16			x	x	x	x	x	x	x	x	
	17	18	19	20	21	22	23	24			x	x	x	x	x	x	x	x	
	25	26	27	28	29	30	31	32			x	x	x	x	x	x	x	x	
	33	34	35	36	37	38	39	40			x	x	x	x	x	x	x	x	
	41	42	43	44	45	46	47	48			x	x	x	x	x	x	x	x	
	49	50	51	52	53	54	55	56			x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	$]$		1	2	3	4	5	6	7	8	$]$

"x" means that "nothing" is injected at that place

Table 2: the injections  $I_1$  (56  $\rightarrow$  64) and  $I_2$  (8  $\rightarrow$  64)



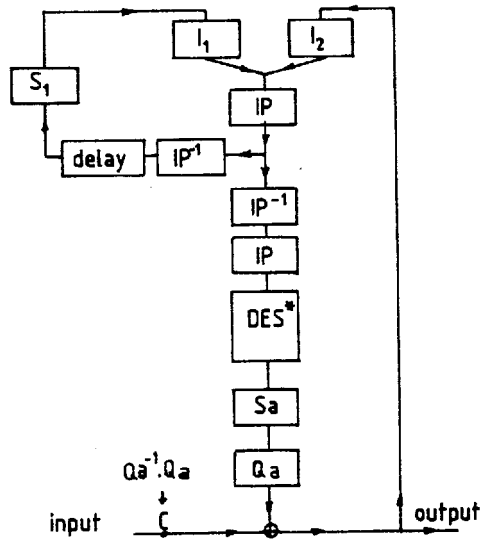


Figure 5: intermediate result of the CFB 1 byte mode derived from figure 4

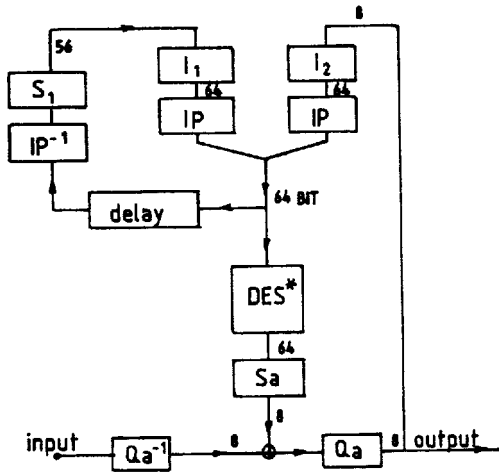


Figure 6: intermediate result of the CFB 1 byte mode derived from figure 5

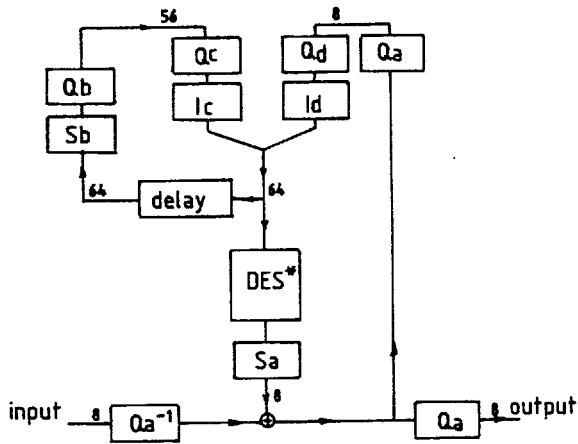


Figure 7: intermediate result of the CFB 1 byte mode derived from figure 6

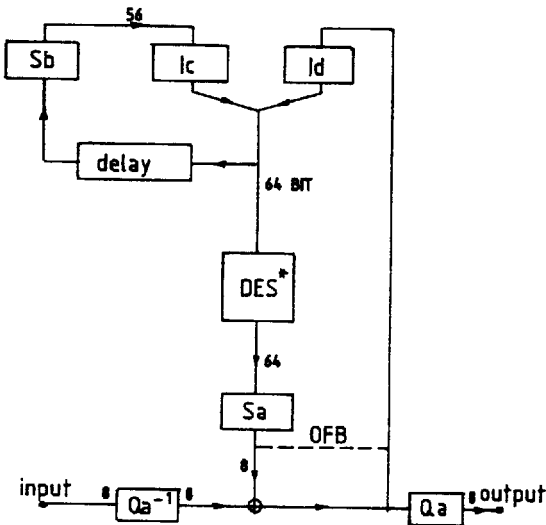


Figure 8: final result of the CFB 1 byte mode derived from figure 7 without permutations in the feedback loop

$$S_a = [ \quad 8 \quad 16 \quad 24 \quad 32 \quad 40 \quad 48 \quad 56 \quad 64 \quad ]$$

$$S_b = [ \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 & \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & \\ 49 & 50 & 51 & 52 & 53 & 54 & 55 & \\ 57 & 58 & 59 & 60 & 61 & 62 & 63 & \end{array} ]$$

Table 3: the selections  $S_a$  ( $64 \rightarrow 8$ ) and  $S_b$  ( $64 \rightarrow 56$ )

$$Q_a = [ \quad 5 \quad 1 \quad 6 \quad 2 \quad 7 \quad 3 \quad 8 \quad 4 \quad ] \quad Q_a^{-1} = [ \quad 2 \quad 4 \quad 6 \quad 8 \quad 1 \quad 3 \quad 5 \quad 7 \quad ]$$

Table 4: the permutations  $Q_a$  ( $8 \rightarrow 8$ ) and  $Q_a^{-1}$  ( $8 \rightarrow 8$ )

$$Q_b = [ \begin{array}{cccccccc} 35 & 7 & 42 & 14 & 49 & 21 & 56 & 28 \\ 34 & 6 & 41 & 13 & 48 & 20 & 55 & 27 \\ 33 & 5 & 40 & 12 & 47 & 19 & 54 & 26 \\ 32 & 4 & 39 & 11 & 46 & 18 & 53 & 25 \\ 31 & 3 & 38 & 10 & 45 & 17 & 52 & 24 \\ 30 & 2 & 37 & 9 & 44 & 16 & 51 & 23 \\ 29 & 1 & 36 & 8 & 43 & 15 & 50 & 22 \end{array} ]$$

$$Q_c = [ \begin{array}{cccccccc} 50 & 42 & 34 & 26 & 18 & 10 & 2 & \\ 52 & 44 & 36 & 28 & 20 & 12 & 4 & \\ 54 & 46 & 38 & 30 & 22 & 14 & 6 & \\ 56 & 48 & 40 & 32 & 24 & 16 & 8 & \\ 49 & 41 & 33 & 25 & 17 & 9 & 1 & \\ 51 & 43 & 35 & 27 & 19 & 11 & 3 & \\ 53 & 45 & 37 & 29 & 21 & 13 & 5 & \\ 55 & 47 & 39 & 31 & 23 & 15 & 7 & \end{array} ]$$

$$Q_d = [ \quad 2 \quad 4 \quad 6 \quad 8 \quad 1 \quad 3 \quad 5 \quad 7 \quad ]$$

Table 5: the permutations  $Q_b$  ( $56 \rightarrow 56$ ),  $Q_c$  ( $56 \rightarrow 56$ ) and  $Q_d$  ( $8 \rightarrow 8$ )

$I_c = [$	x	1	2	3	4	5	6	7		$I_d = [$	1	x	x	x	x	x	x	x
	x	8	9	10	11	12	13	14			2	x	x	x	x	x	x	x
	x	15	16	17	18	19	20	21			3	x	x	x	x	x	x	x
	x	22	23	24	25	26	27	28			4	x	x	x	x	x	x	x
	x	29	30	31	32	33	34	35			5	x	x	x	x	x	x	x
	x	36	37	38	39	40	41	42			6	x	x	x	x	x	x	x
	x	43	44	45	46	47	48	49			7	x	x	x	x	x	x	x
	x	50	51	52	53	54	55	56	]		8	x	x	x	x	x	x	x

Table 6: the injections  $IC$  ( $56 \rightarrow 64$ ) and  $ID$  ( $8 \rightarrow 64$ )

$I_d(Q_d)$ . By calculating all these tables we can proof that  $Q_c(Q_b) = I$  and  $Q_d(Q_a) = I$ , with  $I$  the identity permutation.

In figure 8 we find the final result. We see that  $IP$  is no longer used in the feedback loop, and that two  $8 \rightarrow 8$  permutations have appeared at the input and output of the data. The content of the new permutations, injections and selections is showed in table 3, 4, 5 and 6. They must be read as in the NBS representation [9].

A similar result for the OFB mode can be obtained using similar techniques as for the CFB mode.

Remark that this equivalent representation of the modes can also speed up the software, as explained in [14].

### 3. a fast serial/parallel realization for the permutations

The transport part of the chip will communicate with the environment using 8 bit buses. The main reason for this is the limitation on the amount of connection pins of the chip. However these buses allow us also to realize the permutations  $IP$ ,  $IP^{-1}$  and  $PC_1$  in an elegant way.

The idea is that by shifting data from a bus into a shiftregister, you carry out a permutation in a hidden way. If you put an  $8 \rightarrow 8$  permutation between the bus and the shiftregister (see figure 9), you can realize a whole set of permutations. We found that  $IP$ ,  $IP^{-1}$  and  $PC_1$  can be realized using this set of permutations.

#### 3.1. permuting with shiftregisters: an introduction

When you send a block of 64 bit over an 8 bit bus, you'll send it byte after byte. If we place at the end of the bus 8 shiftregisters of each 8 bit, we can enter sequentially these 8 bytes ( $8 \times 8$ ) (see figure 10). Normally, we'll call the first shiftregister the first byte of our memory, the second shiftregister the second byte etc. (indicated with italic numbers e.g. 1 2).

What we see is that the numbers of the memory locations (1, 2, 3, ..., 64) and the numbers of the data bits (1, 2, 3, ..., 64) don't agree. Conclusion, we have carried out a permutation. This permutation is represented by the vector at the foot of figure 10

(for the interpretation of the vector notation, we refer to the Fibs publication of the data encryption standard [9]). From now on we'll call this permutation  $SR$ .

In the same way, we can show (see figure 11) that when we read out information of 8 shiftregisters into a bus, we carry out another permutation. You can easily check that we get in that way the permutation  $SR^{-1}$ .

Now that we know  $SR$  and  $SR^{-1}$ , we'll search for which 8→8 permutations we need to realize  $IP$ ,  $IP^{-1}$  and  $PC_1$ .

### 3.2. realization of $IP$ and $IP^{-1}$

If we want to have  $IP$  after we have shifted in, we have to do a 64→64 permutation before  $SR$ , satisfying the following equation:

$$IP = SR \text{ after } IP^* \quad \text{or} \quad IP^* = SR^{-1} \text{ after } IP$$

The result is shown in table 7. If we write e.g. bit 18 as the 2' bit of byte 3 (or  $2_3$ ) we see that  $IP^*$  can be realized by carrying out on each byte the same 8→8 permutation  $IP^{**}$  (see table 8).

So the realization of  $IP$  has become very simple as shown in figure 12. This realization is much smaller than a hardware connection path realizing the permutation. At the same time, the execution of  $IP$  doesn't consume extra time because it is carried out during the input from the environment.

A similar method is used to realize  $IP^{-1}$ .

$$IP^{-1} = IP^{-1*} \text{ after } SR^{-1} \quad \text{or} \quad IP^{-1*} = IP^{-1*} \text{ after } SR$$

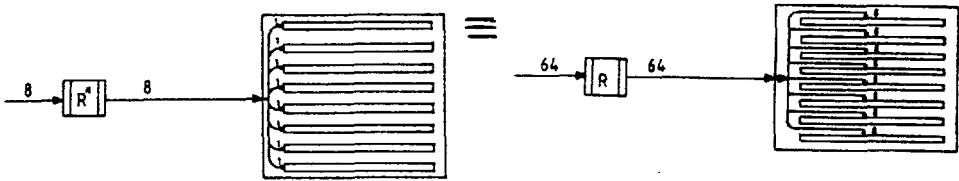
Note that it is possible to use the same shiftregisters to carry out  $IP$  and  $IP^{-1}$ . This can be done simultaneously by reading 1 byte out every time you read 1 byte in (see figure 12).

### 3.3. realization of $PC_1$

For  $PC_1$  we have a more complicated solution because of the irregularity in  $PC_1$  [3]. This can be shown by rewriting the notation of  $PC_1$  (table 9). If we could turn the second part of the permutation  $PC_1$ , we get the permutation  $SR$  (for 7 shiftregisters of 1 byte). This is exactly the way we will realize the permutation.

First we realize  $SR(7 \times 8)$  with 7 shiftregisters during the input of the key in a similar way as for  $IP$  (see the ——— path in figure 13). Then we rearrange these seven registers in two registers of 28 bit. The first register of 28 bit goes from the first byte up to half the fourth byte. The second register goes in reversed order from the last byte up to half the fourth byte (see the - - - - - path in figure 13). This reversed order of the second 28 bit register permits to turn the second part of the key at the moment that those 2 registers of 28 bit are loaded in a following memory unit of 2 times 28 bit (see figure 13).

This realization consumes a little bit more time, but this isn't a drawback because you don't change the key very often. A variant is possible which change the 2 keys e.g. in 1 clockperiod but this consumes a little bit more place. The fact of using 2 memory-units for 2 keys can be used for the multiple key mode. We'll take the first memory as the major-key register and the second register as the active-key register. Therefore we only have to add 2 feedbacklines which carry back the content of the active-key register to the



$R^*$  and  $R$  are permutations

Figure 9: With this shiftregister structure it is possible to carry out a set of  $64 \rightarrow 64$  permutations.

$$IP^* = \begin{bmatrix} 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 \\ 10 & 12 & 14 & 16 & 9 & 11 & 13 & 15 \\ 18 & 20 & 22 & 24 & 17 & 19 & 21 & 23 \\ 26 & 28 & 30 & 32 & 25 & 27 & 29 & 31 \\ 34 & 36 & 38 & 40 & 33 & 35 & 37 & 39 \\ 42 & 44 & 46 & 48 & 41 & 43 & 45 & 47 \\ 50 & 52 & 54 & 56 & 49 & 51 & 53 & 55 \\ 58 & 60 & 62 & 64 & 57 & 59 & 61 & 63 \end{bmatrix}$$

Table 7:  $IP^* = SR^{-1}$  after  $IP$

$$IP^* = \begin{bmatrix} 2_1 & 4_1 & 6_1 & 8_1 & 1_1 & 3_1 & 5_1 & 7_1 \\ 2_2 & 4_2 & 6_2 & 8_2 & 1_2 & 3_2 & 5_2 & 7_2 \\ 2_3 & 4_3 & 6_3 & 8_3 & 1_3 & 3_3 & 5_3 & 7_3 \\ 2_4 & 4_4 & 6_4 & 8_4 & 1_4 & 3_4 & 5_4 & 7_4 \\ 2_5 & 4_5 & 6_5 & 8_5 & 1_5 & 3_5 & 5_5 & 7_5 \\ 2_6 & 4_6 & 6_6 & 8_6 & 1_6 & 3_6 & 5_6 & 7_6 \\ 2_7 & 4_7 & 6_7 & 8_7 & 1_7 & 3_7 & 5_7 & 7_7 \\ 2_8 & 4_8 & 6_8 & 8_8 & 1_8 & 3_8 & 5_8 & 7_8 \end{bmatrix}$$

$$IP^{**} = \begin{bmatrix} 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 \end{bmatrix}$$

Table 8:  $IP^* \equiv 8$  times  $IP^{**}$

the input is:

byte1=	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
byte2=	9	10	11	12	13	14	15	16
byte3=	17	18	19	20	21	22	23	24
byte4=	25	26	27	28	29	30	31	32
byte5=	33	34	35	36	37	38	39	40
byte6=	41	42	43	44	45	46	47	48
byte7=	49	50	51	52	53	54	55	56
byte8=	57	58	59	60	61	62	63	64

the result is:

→	<b>57</b>	<i>1</i>	<b>49</b>	<i>2</i>	<b>41</b>	<i>3</i>	<b>33</b>	<i>4</i>	<b>25</b>	<i>5</i>	<b>17</b>	<i>6</i>	<b>9</b>	<i>7</i>	<b>1</b>	<i>8</i>
→	<b>58</b>	<i>9</i>	<b>50</b>	<i>10</i>	<b>42</b>	<i>11</i>	<b>34</b>	<i>12</i>	<b>26</b>	<i>13</i>	<b>18</b>	<i>14</i>	<b>10</b>	<i>15</i>	<b>2</b>	<i>16</i>
→	<b>59</b>	<i>17</i>	<b>51</b>	<i>18</i>	<b>43</b>	<i>19</i>	<b>35</b>	<i>20</i>	<b>27</b>	<i>21</i>	<b>19</b>	<i>22</i>	<b>11</b>	<i>23</i>	<b>3</b>	<i>24</i>
→	<b>60</b>	<i>25</i>	<b>52</b>	<i>26</i>	<b>44</b>	<i>27</i>	<b>36</b>	<i>28</i>	<b>28</b>	<i>29</i>	<b>20</b>	<i>30</i>	<b>12</b>	<i>31</i>	<b>4</b>	<i>32</i>
→	<b>61</b>	<i>33</i>	<b>53</b>	<i>34</i>	<b>45</b>	<i>35</i>	<b>37</b>	<i>36</i>	<b>29</b>	<i>37</i>	<b>21</b>	<i>38</i>	<b>13</b>	<i>39</i>	<b>5</b>	<i>40</i>
→	<b>62</b>	<i>41</i>	<b>54</b>	<i>42</i>	<b>46</b>	<i>43</i>	<b>38</b>	<i>44</i>	<b>30</b>	<i>45</i>	<b>22</b>	<i>46</i>	<b>14</b>	<i>47</i>	<b>6</b>	<i>48</i>
→	<b>63</b>	<i>49</i>	<b>55</b>	<i>50</i>	<b>47</b>	<i>51</i>	<b>39</b>	<i>52</i>	<b>31</b>	<i>53</i>	<b>23</b>	<i>54</i>	<b>15</b>	<i>55</i>	<b>7</b>	<i>56</i>
→	<b>64</b>	<i>57</i>	<b>56</b>	<i>58</i>	<b>48</b>	<i>59</i>	<b>40</b>	<i>60</i>	<b>32</b>	<i>61</i>	<b>24</b>	<i>62</i>	<b>16</b>	<i>63</i>	<b>8</b>	<i>64</i>

the realized permutation

$SR = [$	<b>57</b>	<b>49</b>	<b>41</b>	<b>33</b>	<b>25</b>	<b>17</b>	<b>9</b>	<b>1</b>
	<b>58</b>	<b>50</b>	<b>42</b>	<b>34</b>	<b>26</b>	<b>18</b>	<b>10</b>	<b>2</b>
	<b>59</b>	<b>51</b>	<b>43</b>	<b>35</b>	<b>27</b>	<b>19</b>	<b>11</b>	<b>3</b>
	<b>60</b>	<b>52</b>	<b>44</b>	<b>36</b>	<b>28</b>	<b>20</b>	<b>12</b>	<b>4</b>
	<b>61</b>	<b>53</b>	<b>45</b>	<b>37</b>	<b>29</b>	<b>21</b>	<b>13</b>	<b>5</b>
	<b>62</b>	<b>54</b>	<b>46</b>	<b>38</b>	<b>30</b>	<b>22</b>	<b>14</b>	<b>6</b>
	<b>63</b>	<b>55</b>	<b>47</b>	<b>39</b>	<b>31</b>	<b>23</b>	<b>15</b>	<b>7</b>
	<b>64</b>	<b>56</b>	<b>48</b>	<b>40</b>	<b>32</b>	<b>24</b>	<b>16</b>	<b>8</b> ]

the *italic* number indicates a memory location and the **boldface** number a data bit

Figure 10: If you shift 8 byte from a bus into 8 shiftregisters, you get the permutation  $SR$  (a permutation is represented in a similar way as in the NBS norm of DES)

1	2	3	4	5	6	7	8	→
9	10	11	12	13	14	15	16	→
17	18	19	20	21	22	23	24	→
25	26	27	28	29	30	31	32	→
33	34	35	36	37	38	39	40	→
41	42	43	44	45	46	47	48	→
49	50	51	52	53	54	55	56	→
57	58	59	60	61	62	63	64	→

the output is:

byte1=8	16	24	32	40	48	56	64
byte2=7	15	23	31	39	47	55	63
byte3=6	14	22	30	38	46	54	62
byte4=5	13	21	29	37	45	53	61
byte5=4	12	20	28	36	44	52	60
byte6=3	11	19	27	35	43	51	59
byte7=2	10	18	26	34	42	50	58
byte8=1	9	17	25	33	41	49	57

the realized permutation

$SR^{-1} =$	8	16	24	32	40	48	56	64
	7	15	23	31	39	47	55	63
	6	14	22	30	38	46	54	62
	5	13	21	29	37	45	53	61
	4	12	20	28	36	44	52	60
	3	11	19	27	35	43	51	59
	2	10	18	26	34	42	50	58
	1	9	17	25	33	41	49	57

Figure 11: by reading out of a shiftregister you realize the permutation  $SR^{-1}$



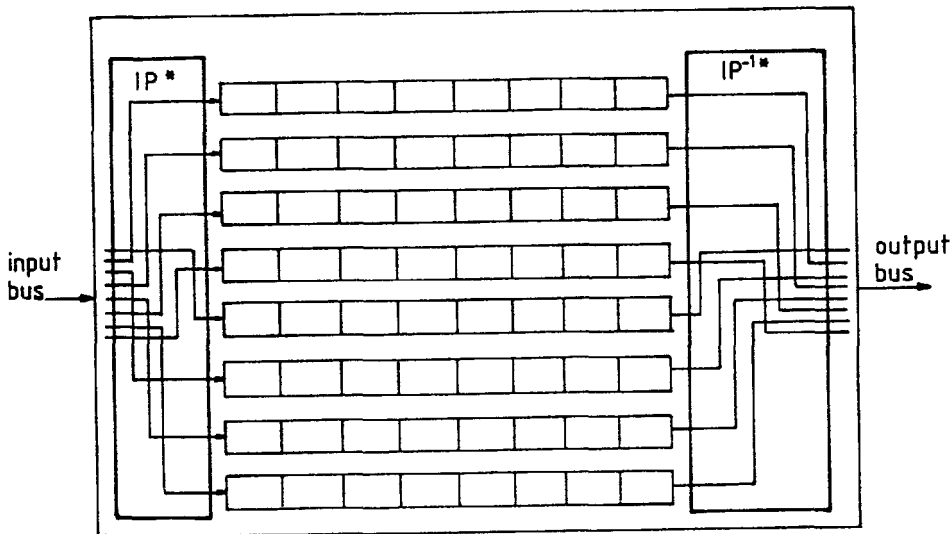


Figure 12: realization of  $IP$  and  $IP^{-1}$  with the same shiftregister

major-keyregister so that no key is lost by the transport from the major-key register to the active-key register (see figure 13). This configuration can also be expanded with a third keyregister which is very interesting for multiple encipherment of the active key [12]

#### 4. a fast serial/parallel realization for memory and transport

We'll explain our internal interconnection system built out of shiftregisters and multiplexers. This system is small, very fast and flexible enough for the DES-algorithm.

$PC_1 =$									$SR(7 \times 8) =$							
[ 57	49	41	33	25	17	9	1	$\rightarrow$	[ 57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2		58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3		59	51	43	35	27	19	11	3
60	52	44	36						60	52	44	36	28	20	12	4
63	55	47	39	31	23	15	7		61	53	45	37	29	21	13	5
62	54	46	38	30	22	14	6		62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5		63	55	47	39	31	23	15	7 ]
				28	20	12	4 ]									

Table 9: the structure in  $PC_1$  and the relation with  $SR$

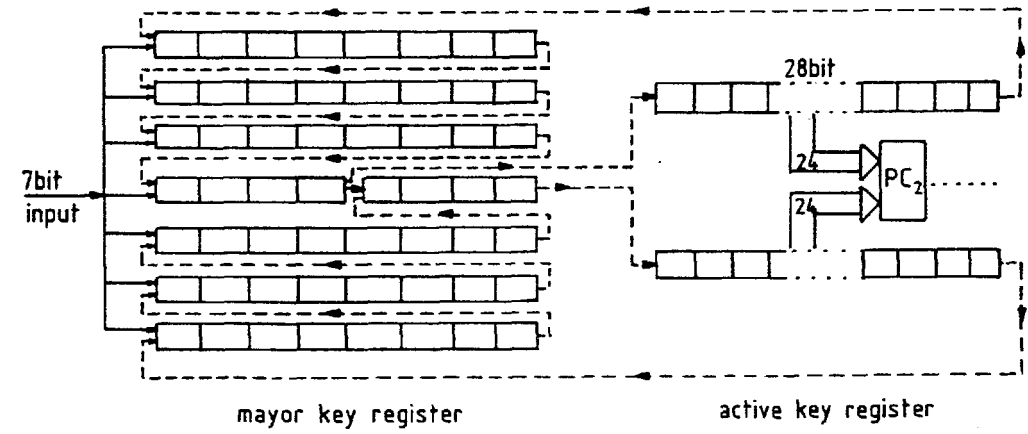
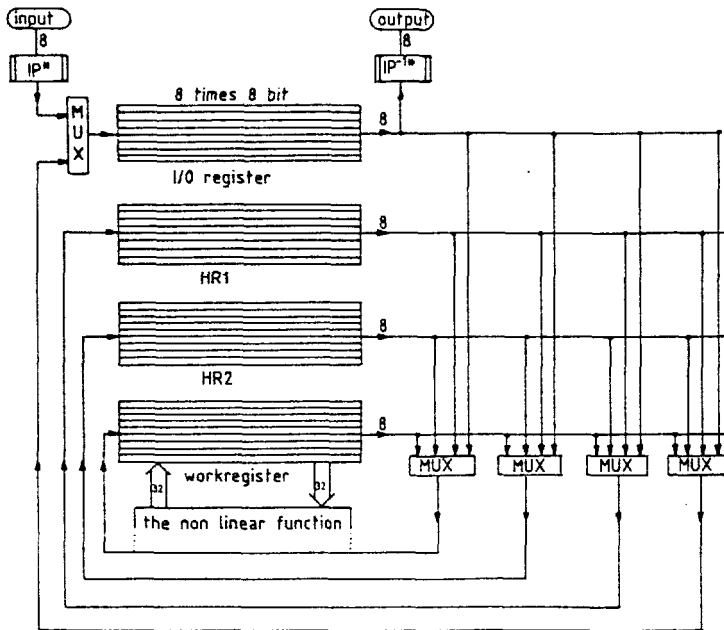


Figure 13: realization of  $PC_1$  by 2 subsequent memory transports



HR1 and HR2 are additional registers. The workregister is the memory unit in the iteration hardware (see section 9).

Figure 14: the fast internal transport organization. It's used to calculate the modes (the exors are omitted from the figure for simplicity)

A common way to organize transport is with one databus and an addressbus. Then the memory usually consists out of RAM and ROM units. The advantage of this solution is the flexibility of addressing and the possibility of transport between many devices far from each other.

We took another approach because we don't need those advantages. We chose to organize the memory in 4 shiftregister units of 8 byte ( $8 \times 8$ ) each. Instead of one bus, we made a connection path from every output to each of the four inputs (included his own input) (see figure 14). For our design this structure is faster, small and enough flexible.

It's faster because of two reasons. First, it isn't necessary to specify a new address for every byte. Second, it can transport 4 bytes simultaneously. So the maximum capacity is 32 byte in 8 clockperiods. We'll use this structure to calculate the feedbackmodes. This is a time critical job because it isn't possible to pipeline it with something else (see section 6).

It is difficult to explain why this structure is very small. On the first sight you may think to discover a new routing- problem. This isn't so because the length of the connections can be kept small by a good floorplan. In section 9.2 we describe a floorplan for a nMOS realization. Most interconnections between these registers aren't longer than  $150\mu m$ . It was even possible to place all the interconnections and multiplexers on  $0,5mm^2$  ( $8 \times 8$  version).

This structure has a smaller flexibility because we always have to transport all 8 bytes in the same sequence from one unit to the other. Transporting less will break up every byte by partly shifting it out and partial shifting something else in. It is clear that this partial transport is a "wrong" way to transport entire bytes. But as we see in section 5.1, we can use this "wrong" way of transport to execute the 1 byte modes in a very simple way. Remark that to execute the modes we also need to incorporate some exors in the structure. This can be done in many ways depending on the used technology. One method well suited for nMOS is adding 8 fixed exors between the workregister and the input output register and also 8 fixed exors between the workregister and the HR1 (= additional register 1). The output of those exors are connected to each multiplexer again. It can be shown that this configuration is sufficient to calculate the modes.

#### 4.1. a possible modification for smaller, but slower devices

Up to now, we've used units of  $8 \times 8$  shiftregisters for the input output registers and the other registers of the fast internal transport. However it's also possible to construct an equivalent version with 4 shiftregisters of 16 bits long [5].

The difference between the two is that the interconnection hardware is only half of the size for the  $4 \times 16$  version. On the other hand the  $4 \times 16$  version is also 2 times slower for the transport of data. Therefore the  $4 \times 16$  version is only usefull for a slow and small version ( $\leq 5$  Mbit and  $\pm 4mm^2$ ).

## 5. Equivalent representations

### 5.1. modes

As explained in section 3.2, we can easily realize  $IP$  when we transport data from an 8 bit bus into a shiftregister. This consumes no extra time when it can be combined with the input and output from and to the environment. However, this method needs a lot of time when you have to use it for data already on chip. Therefor it's very usefull to move the permutations  $IP$  and  $IP^{-1}$  out of the feedback loop of the modes to the input and output of the chip.

Figure 3 shows clearly that for the 8 byte modes the solution is found. However for the 1 byte modes, a little bit more explanation is needed to show why the result in figure 8 is usefull. The permutations  $Q_a$  and  $Q_a^{-1}$  of figure 8 are the permutations  $IP^*$  and  $IP^{-1*}$  we used to read in and out (section 3). The realization of the selections and injections is very simple with our internal transport structure (section 4). We saw there that you can only transport entire bytes if you transport all 8 bytes together by shifting 8 times the shiftregisters. However if you shift those registers only once, you will eject the last bit of every byte ( $=S_a$ ). The other seven bits of every byte ( $=S_b$ ) will be shifted to the seven last places of every byte ( $=I_c$ ) and there will enter a new bit on the first place of every byte ( $=I_d$ ). This new bit is the ejected bit exored with the correspondent bit of the incoming byte of data.

### 5.2. the permutation P

Because the permutation P will be realized hardware with wires, it's obvious that the needed area can be diminished by a well choosen modified P [3]. How you get the optimal

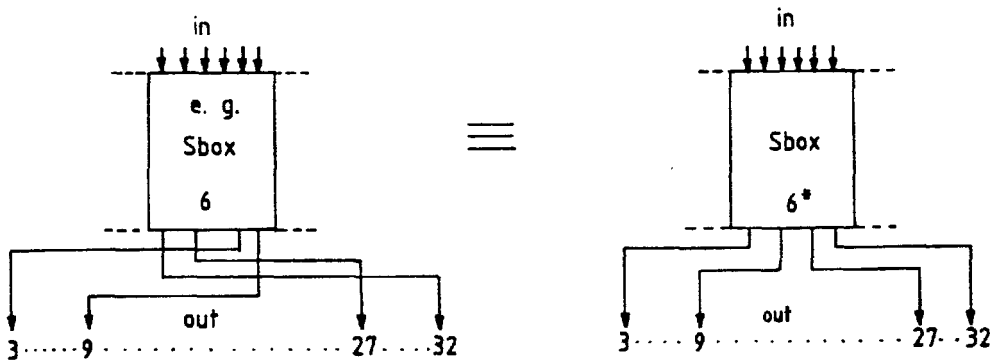


Figure 15: optimization of P for a hardwired realization

modified P is shown in figure 15.

### 5.3. optimization of the 16 iterations

Starting from the official representation of the DES algorithm, we see that each iteration starts with the evaluation of the non-linear function, and ends with reversing the 2 datablocks. If you reorder [13] the algorithm as in figure 16, it becomes clear that the exchange can be done at the same moment as the evaluation of the non-linear function. In this way, the time to execute DES\* (=DES without  $IP$  and  $IP^{-1}$ ) is almost equal to the time needed for 16 evaluations without losing time by exchanging the 2 datablocks. As consequence, the evaluation of the non-linear function is the time critical path of the algorithm. The only small drawback is that at the end an extra exchange of the right and left block is required.

## 6. Pipelining

The aim of the pipelining is to prevent that datarate is slowed down by the time used for datatransport between chip and the environment. This is very important because this communication is slow.

In our design we distinct 3 sections able to work simultaneously: an input section, a DES\* section and an output section. While the input section is entering the next datablock, the DES\* is working on the actual datablock and the outputsection is busy to releaze the previous datablock. When these 3 sections have finished, there is a large amount of data which has to be exchanged between the sections. The operation doing this job is called the transfert. The transfert is executed with the structure described in section 4 . This operation calculates also the modes.

To illustrate the functioning of the device, especially to show how the modes are realized, we have in figure 17 a representation of the activity in function of time e.g. for the modes ECB and CBC. Note that HR1 and HR2 (additional memory 1 and 2) are memories needed in the feedbackloop of the modes.

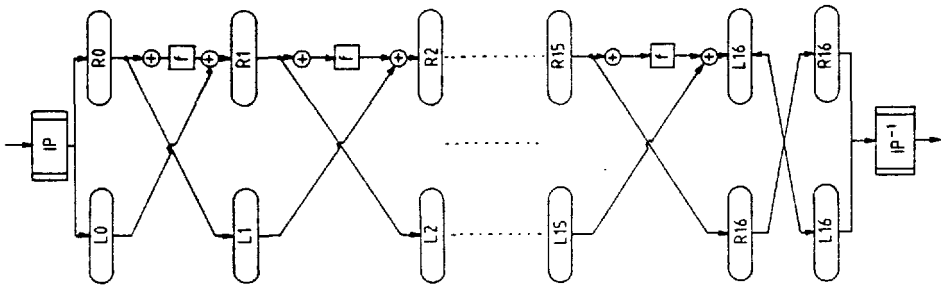


Figure 16: optimization of the 16 iterations

## 7. Rearrangement of the functional elements

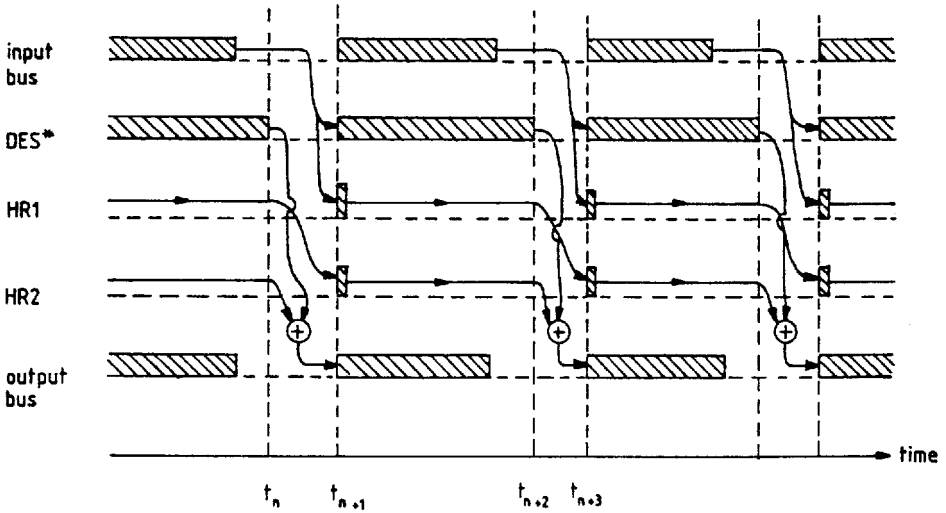
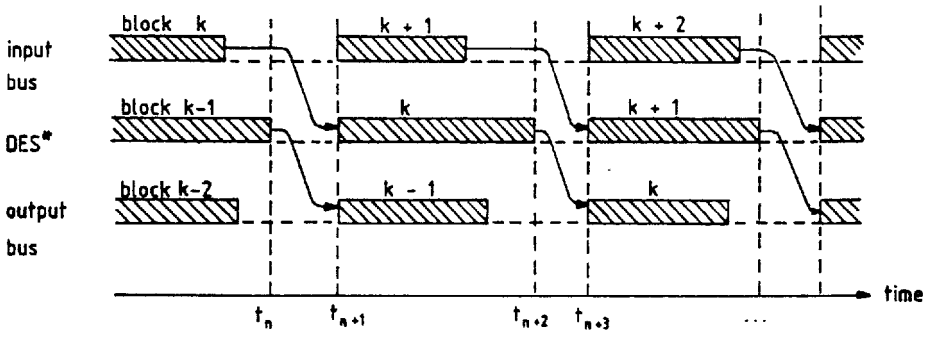
The aim of rearranging is not to avoid the interconnections but to shorten the length.


A large part of the routing is shortened by mixing the memory cells in the iteration hardware. If you number the cells from 1 to 64 and put them in the following order: (1 33), (2 34), (3 35), (4 36), ... (29 61), (30 62), (31 63), (32 64), the connected cells come next to each other. So the connections become shorter than  $100\mu m$ . This structure can be build with 32 cells each containing a schiftregister of 2 bit and 1 exor.


The second way to shorten lines relies on the fact that a memorycell is much larger than an interconnection. So we'll put the lines from the subkey, the lines from the memory and the lines from the S-boxes wired next to each other. In this way we have designed a floorplan for the iteration hardware that minimize the length between the memory and the S-boxes that is part of the time critical path (cfr. section 5.3).

## 8. Modular construction of the controller

Microprogramming is known as a good but slow way of controlling. The speed problem can be solved by using a lot of small units. Every unit is able to carry out one class of tasks. Above this units for the tasks is one unit to coordinate the cooperation between the units. The communication between this coordination unit and the task units happens with microcode. There is no communication between the task units so that modularity and testability is assured.



 :shows that this part is active

 :shows the transfert of data between the different memories.

Remark for CBC-decipher that the calculation of DES\* introduces a supplementary delay of the data in the main path so that in the feedback path 2 delays are needed instead of 1.

Figure 17: the activity in function of time for the modes ECB and CBC-decipher

## 9. The entire design

### 9.1. the architecture

In figure 18 the survey of the total architecture is shown. Depending on the design strategy, used technology and the desired features some elements may have to change. The solution of the figure is very well suited for an nMOS chip with a speed of about 14Mbit/sec. .

As seen in section 4, there are 4 data memories of 64 bit ( $=8 \times 1$  byte). The memory called workregister is a special one because it can work in 2 different ways. It can be switched as 8 shiftregisters of 1 byte, or as 32 registers of 2 bit to perform the 16 iterations (see section 7).

There are also 2 key memories of 56 bit ( $=7 \times 1$  byte). The parity check is carried out on each key byte that is entered for the first time. With this configuration you can memorize 2 keys e.g. a major and an active key. When you enter a new key, you'll overwrite the key which was in the input key register.

In the chip we have the following three transports:

1. the internal transport, mainly used for realizing the modes
2. an inputbus
3. an outputbus

The first serves to transport data in a fast way between the 4 data memories (see section 4). The second and the third serve for datatransport between the environment, one fixed register of 64 bit (called the input output register) and the active key register (see figure 12 and 13).

Maybe it is now a good moment to take attention on the conformity of the used techniques. E.g. reading in the data needs shiftregisters to realize at the same time *IP*; with those shiftregisters a very fast transport is possible; we need that fast transport to allow the pipelining and the execution of the modes; to allow this way of transport, the workregister has to be a shiftregister; on the other hand, the iterations can be carried out very fast using cells of a shiftregister,... etc.. It is with this conformity that we could design a chip which at the same time is very fast and very small.

### 9.2. the floorplan

In figure 19 you find a floorplan of an nMOS design. The total area used is about 9  $mm^2$  and the transistor density is more than a thousand transistors on 1  $mm^2$ . In the floorplan you can distinct 4 important parts.

1. The datapad doing the 16 iterations (+ the subkey generation)
2. The memories and multiplexers of the fast internal transport
3. The input-output bus
4. The controllers





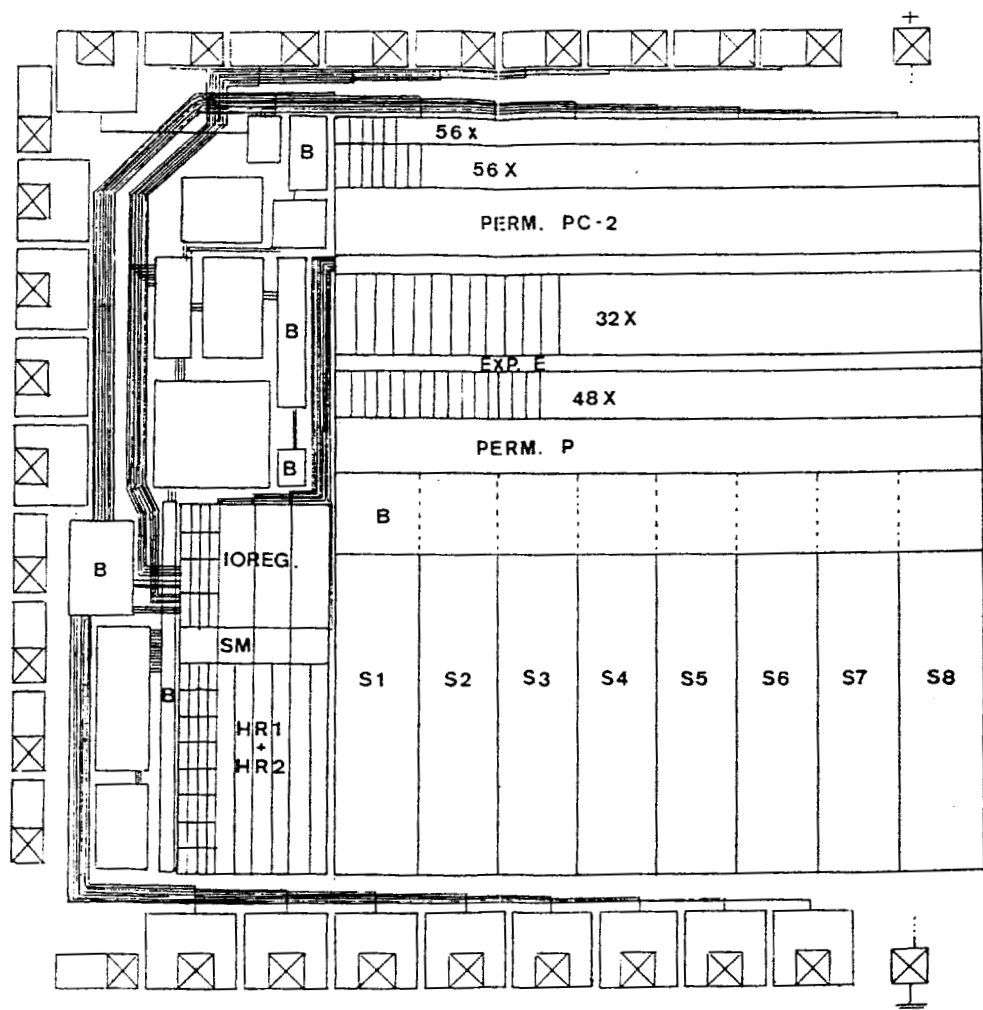


Figure 19: floorplan for a nMOS version

## 10. a modified design for exhaustive search of the keys

The idea to cryptanalyze the DES by an exhaustive search was proposed by Diffie and Hellman [15]. An improvement is now presented, which mainly solves the problem of the complexity of the machine and its cooling.

A chip builded for exhaustive search of the keys must have 2 properties. First it has to be very fast and second it must work with a minimum of communication with the environment.

To make it fast we'll use the property that an exhaustive search of the key can be realized using only the ECB mode. Therefore we will divide the path calculating the non-linear function in e.g. 3 section and pipeline those sections. There is however the problem that we need the result of this non-linear function to calculate the new input for the non-linear function. This will be solved by calculating simultaneous DES for 3 different keys. To do that we need three workregisters and three keyregisters. In that way the speed can be improved by a factor 3.

To minimize the communication with the environment, we'll generate the subsequent keys on chip and we'll do the check of the result also on chip. To generate the subsequent keys, we enter once a start value for the key in a counter on chip and then augment this value each time by 1. By giving each chip a good startvalue, the whole key space will be checked. To check the result, we enter only once the 2 datablocks for which we search the key. The first block will serve as input for the DES algorithm and the result of the DES algorithm will be compared with the second datablock on chip. If the result is equal to the second block, an interrupt signal is given. In this way, only a  $\mu$ computer and a big power supply is needed to command e.g. 10,000 chips.

*Important is at which speed this device can work!* To calculate three outputs you need:

$48 (=16 \text{ iterations}) + 2 (= \text{delay in the pipeline}) + 3 (= \text{time for in and output}) = 53 \text{ clockcycli.}$

At a clockfrequency of 20 Mhz you can check  $\pm 1.13 \cdot 10^6$  keys in one second. Suppose that one device (incl. connection) costs 40\$ and that you spend 1,000,000 \$, you can with  $2.5 \cdot 10^4$  devices calculate  $2.8 \cdot 10^{10}$  keys in one second, or  $1 \cdot 10^{14}$  keys in one hour. So you calculate  $1.7 \cdot 10^{16}$  keys in only 1 week. In total there are  $7.2 \cdot 10^{16}$  keys. On the average you will find the key after you've tried  $3.6 \cdot 10^{16}$  keys and for this you need about two weeks. If a chosen plaintext attack is possible, the time needed to find the key is divided by 2 [11]. It may also be necessary to make allowance for more than one key satisfying:  $\text{cypherblock}(64\text{bit}) = \text{DES}(\text{plaintext}, \text{key})$  [7].

The proposed hardware can be designed for CMOS such that no power problems would exist.

## 11. Obtained results

Without exaggerating, we may expect that a speed up to 20 Mbit/sec. is possible to obtain. Maybe higher speeds should be possible, but this should certainly cost a large amount of power and area.

It's easy to calculate the speed by hand. Because of the pipelining, the chip is sometimes doing many tasks simultaneously (see figure 17). The chip is so designed that the

clock	datarate	area
3Mhz	3,4Mbit	2x2 mm <sup>2</sup>
5Mhz	5,5Mbit	
10Mhz	11 Mbit	3x3 mm <sup>2</sup>
14Mhz	15,5Mbit	
18Mhz	20 Mbit	4x4 mm <sup>2</sup>

Table 10: datarate in function of the clockfrequency (with the expected area in  $3\mu\text{m}$  technology)

evaluation of the 16 iterations takes normally most of the time. So the time needed for encrypting or decrypting each datablock is the time for 16 iterations augmented with the time for the transfert. In our design, 1 iteration needs 3 clockcycli. Finally, we also need 2 clockcycli to allow the controller to jump from one to an other task unit (section 8). Together we get:

	Transfert: 8 clockperiods
DES*	:48 clockperiods
jumps	: 2 clockperiods
Total	:58 clockperiods

This means 64/58 bits in one clockperiod. To know the total speed you have to estimate the clockfrequency. In our design there are 2 to 3 gatelevels for every  $1/2$  clockperiod (2 phase clock)[10]. So a high clockspeed can surely be obtained. In table 10 we give some frequencies and the corresponding speed. We also indicate the possibility to exchange speed versus area (and power). One can agree that the design is very compact compared with his performances. Smaller technology should futher minimize the area.

## 12. Conclusions

In this paper we presented several improvement in order to realize faster and smaller chips.

We also proved that the initial permutation and the inverse initial permutation can always be located at the input, respectively the output of each mode.

To use DES in a strong way one has to change frequently the active key (e.g. every 10 seconds) and this active key must be multiple enciphered with different major keys before transmission.

## Acknowledgment

The authors are grateful to Dr. J.-J. Quisquater and Prof. M. Davio (Philips Research Laboratory, Brussels, Belgium), for discussions about this paper.

- [1] R. Cushman, "Data encryption chips provide security, is it false security," *EDN*, February 1982, pp. 39 - 42.
- [2] A. Konheim, "*Cryptography: A Primer*," John Wiley, Toronto, 1981.
- [3] M. Davio, Y. Desmedt, M. Fosseprez, R. Govaerts, J. Hulsbosch, P. Neutjens, P. Piret, J.-J. Quisquater, J. Vandewalle, P. Wouters, "Analytical Characteristics of the DES," *Advances in Cryptology, Proc. Crypto 83*, August 1983, Plenum, pp. 171 - 202.
- [4] D. Davies, "The Data Encryption Standard," *International Course on Cryptography and Computer Security*, ESAT, K.U.Leuven, Belgium, November 29 - December 2, 1983.
- [5] J. Goubert and F. Hoornaert, "Study about an efficient chip implementation of the DES (in Dutch: Studie van een efficiënte implementatie van het DES algoritme op chip)," Final work, K.U.Leuven, Department Elektrotechniek, June 1984.
- [6] H. De Man, L. Reynders, M. Bartholomeus and J. Cornelissen, "Plasco: A silicon compiler for nMOS and CMOS PLA," *Proc. IFIP: VLSI 83*, Trondheim, Norway, August 1983, pp. 171 - 181.
- [7] Y. Desmedt, J.-J. Quisquater and M. Davio, "Dependence of output on input in DES: Small avalanche characteristics," *Advances in Cryptology, Proc. Crypto 84*, August 1984, Springer-Verlag.
- [8] FIPS publication, "DES modes of operation," *Federal Information Processing Standard*, no.81, National Bureau of Standards, U. S. Department of Commerce, Washington D. C. , U. S. A. , December 2, 1980.
- [9] "Data Encryption Standard," *FIPS* (NBS Federal Information Processing Standards Publ. ), no. 46, January 1977
- [10] C.Mead and Conway, "An introduction to VLSI systems," Addison Wesley, Reading, 1980.
- [11] M. Hellman, R. Merkle, R. Schroepfel, L. Washington, W. Diffie, S. Pohlig and P. Schweitzer, "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard," *Electrical Engineering Dep. Stanford Univ.* , SEL 76-042.
- [12] American National Standard Institute, "Financial Institution Keymanagement" Draft April 1984, N216.
- [13] C. H. Meyer and S. M. Matyas, "*Cryptography: A New Dimension in Computer Data Security*," J. Wiley, New York, 1982.
- [14] M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, and J.-J. Quisquater, "Efficient hardware and software implementations to the DES," Abstract, *Advances in Cryptology, Proc. Crypto 84*, August 1984, Springer-Verlag.
- [15] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, no. 6, pp. 74 - 84, June 1977.

A SELF-SYNCHRONIZING CASCADED CIPHER SYSTEM  
WITH DYNAMIC CONTROL OF ERROR PROPAGATION

Norman Proctor  
SYTEK, Incorporated  
1225 Charleston Road  
Mt. View, California 94039-7225

ABSTRACT

A cipher system used for secure communication over a noisy channel can automatically synchronize the sender and receiver by computing a stateless function of a key and a limited amount of the recent ciphertext. The more ciphertext feedback is used, the more the errors from the noisy channel are propagated. The less feedback is used, the easier ciphertext-only and chosen-plaintext attacks become. There is a trade-off between security and noise that must be made when a self-synchronizing system is built.

This paper presents a self-synchronizing cascaded cipher system that permits most combinations of key and ciphertext feedback lengths and also allows adjustment of the trade-off between security and noise during system operation. At times when maximum security is not needed, the error propagation can be reduced temporarily.

As implemented in hardware, the cascaded cipher has a storage register for each stage. The function computed would normally depend on the state of this storage, but different clocks are used at each stage to render the function stateless. The use of a cascade helps to keep the hardware cost down.

CIPHER FEEDBACK SYSTEMS

Cipher feedback systems are useful in circumstances where synchronization between the encipherer and decipherer cannot be guaranteed in any practical way. Figure 1 shows a generic cipher feedback system.

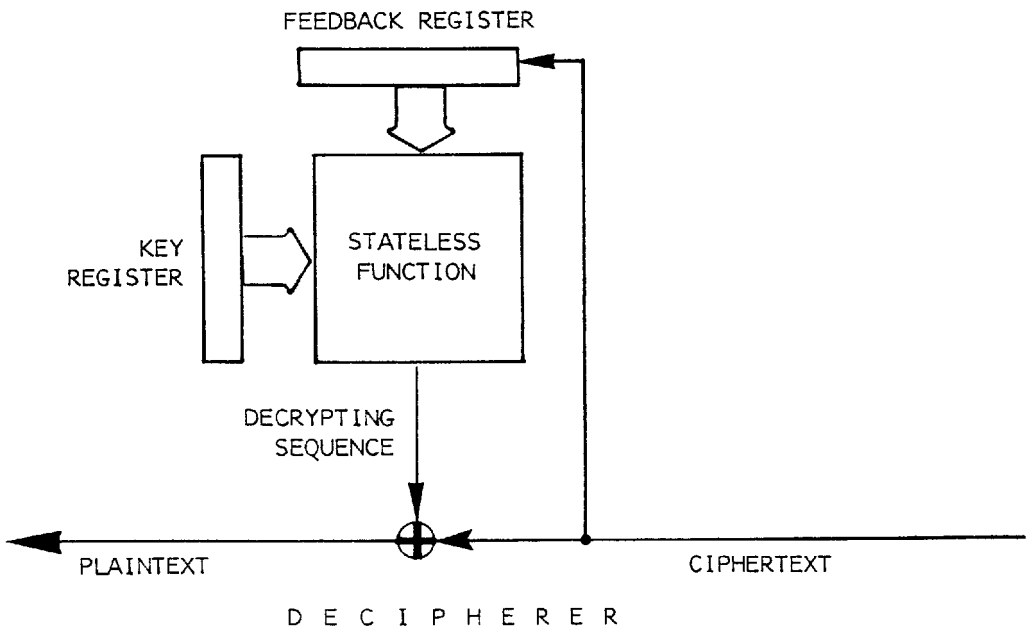
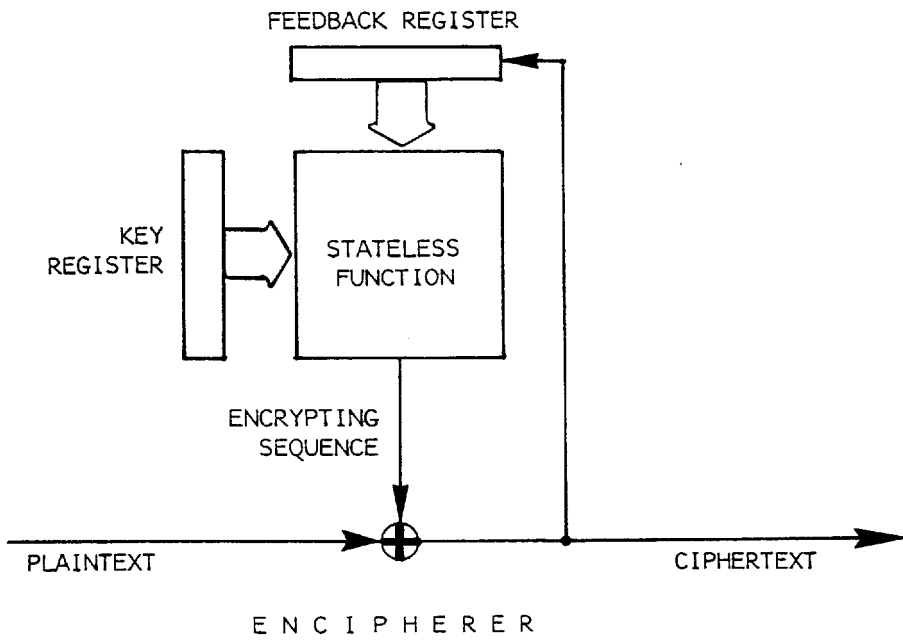
**CIPHER FEEDBACK SYSTEM**

FIGURE 1

The encipherer consists of a  $k$ -bit key register, an  $f$ -bit feedback shift register and a combinatorial circuit with inputs from the two registers and with one output. The sequence of output bits will be called the encrypting sequence. It is a pseudo-random sequence if the combinatorial circuit is correctly chosen. Some constraints may need to be placed on the choice of keys in order to achieve pseudo-randomness. The encrypting sequence is combined with the plaintext sequence by modulo-2 addition to produce the ciphertext sequence. As each bit of the ciphertext sequence is produced, it is shifted into the feedback register. This means that each ciphertext bit may depend on the  $k$  bits of the key, the previous  $f$  bits of the ciphertext and the current plaintext bit.

The decipherer like the encipherer has a  $k$ -bit key register, an  $f$ -bit feedback register and a combinatorial circuit identical to the one in the encipherer. The decrypting sequence from that circuit is combined with the ciphertext sequence to produce a plaintext sequence. After each bit of the ciphertext sequence is first used, it is shifted into the feedback register so that it is also used to produce the next  $f$  bits of the decrypting sequence.

#### SELF-SYNCHRONIZATION

A bit of the decrypting sequence will be the same as the corresponding bit of the encrypting sequence provided only that the encipherer and decipherer are using the same key and that the previous  $f$  bits of ciphertext received by the decipherer are unchanged since the encipherer produced them. We will assume in general that some technique ensures that the encipherer and decipherer do use the same key. The second proviso will not always be met if the channel that carries the ciphertext is noisy for any reason.

Any difference between ciphertext bits produced by the encipherer and the corresponding bits received by the decipherer will be called errors in the ciphertext. Similarly, differences between corresponding bits in the two plaintext sequences or in the encrypting and decrypting sequences will be called errors in the plaintext or in the decrypting sequence.

A bit of the decipherer's plaintext sequence must be the same as the corresponding bit of the encipherer's plaintext sequence provided only that the corresponding encrypting and decrypting sequence bits are the same and the current ciphertext bit received is the same as the current ciphertext bit produced. This means that following a transmis-



sion error which changes some bits in the ciphertext, there can be errors in the plaintext for at most  $f$  bits after the last erroneous ciphertext bit was received. Because the decipherer recovers automatically within a short and fixed number of bits to errors in the ciphertext, cipher feedback systems are said to be self-synchronizing.

Self-synchronization makes it unnecessary for the encipherer and decipherer to agree on a value or time to initialize their feedback registers. The first  $f$  bits of the plaintext sequence are not part of the message because the decipherer cannot be expected to recover them. But barring errors in the ciphertext, the decipherer can recover all the rest of the plaintext sequence no matter what its feedback register held when the first ciphertext bit was received.

Self-synchronization also makes it feasible for the decipherer to interpret the ciphertext stream as a sequence of bits by periodic sampling even though its clock is not exactly as fast as the encipherer's clock. A slight difference in clock speeds would produce occasional bursts of plaintext errors up to  $f + 1$  bits long but would not produce complete gibberish.

There are variants on the system in Figure 1 which retain the self-synchronizing property. The encipherer could use a different invertible computation to produce the ciphertext sequence from the encrypting sequence and the plaintext sequence instead of the self-inverting modulo-2 addition. The decipherer would use the inverse. Another variation has the plaintext and ciphertext processed block by block instead of bit by bit with each ciphertext block (or part of it) being shifted into the feedback registers. We will not explicitly consider such variants, but the discussions below would be analogous or even identical for the variant cipher feedback systems.

The essential features are that (1) the encipherer's and decipherer's feedback registers contain a limited number of bits of the recent ciphertext sequence (or equivalent information), (2) the encrypting and decrypting sequences are produced by the same stateless function of the key and at least some of the contents of the feedback register, and (3) the encrypting sequence is combined with the plaintext in such a way that the plaintext can be recovered from the resulting ciphertext whenever corresponding portions of the encrypting and decrypting sequences match.

## THE SECURITY PROBLEM

The advantages of self-synchronization come with two major disad-

vantages. One is that errors in the ciphertext are propagated. The other is that the length of the feedback register may allow the cipher system to be broken by an exhaustive search less expensive than an exhaustive search of the key space.

As long as the key remains the same, the bit of the encrypting sequence produced after a particular  $f$ -gram in the ciphertext will always be the same. A codebook can be assembled whose index is the values of the previous  $f$  bits of the ciphertext and whose entries are the bits of the encrypting sequence that must follow. It can be used in place of the key and combinatorial circuit in the encipherer or decipherer. The codebook has  $2^f$  bits. If  $2^f$  is less than the key length  $k$ , it is easier to search exhaustively for the right codebook than to search for the right key.

When the codebook has fewer bits than the key, many keys must be equivalent to each other. Each equivalence set of keys is associated with a unique codebook. The mean size of the equivalence sets will be at least 2 raised to  $(k - 2^f)$ . Analysis of the plaintext and ciphertext can identify the key's equivalence set by finding that the set's codebook (or some member of the set) decipheres the ciphertext. But unless the equivalence set turns out to be a singleton set, the key cannot be positively identified from the plaintext and ciphertext. Identifying all the members of the key's equivalence set is more difficult than finding the set's codebook. This fact is useful as we will discuss later if the secrecy of the key can sometimes be more important than the secrecy of the plaintext.

## THE NOISE PROBLEM

A feedback register which is too short relative to the key register can make the cipher feedback system easier to break than it would otherwise be. That makes very short feedback registers undesirable. The error propagation from cipher feedback registers, on the other hand, makes very short feedback registers attractive.

In most other cipher systems, ciphertext errors either are not propagated at all or are potentially propagated indefinitely. If only error-free plaintext were acceptable, indefinite propagation would be preferable as it makes error detection easier. Instead, we will assume that some errors in the plaintext are tolerable but that it is valuable to minimize them. Clearly, the best error propagation is none at all. The error rate in the plaintext would then be no worse than the rate in the ciphertext.

With cipher feedback systems, an error in one bit of the ciphertext is normally propagated within the next  $f$  bits of the plaintext. This is because the decipherer's feedback register holds the erroneous bit while the next  $f$  bits of the decrypting sequence are produced. Any of those  $f$  bits in the decrypting sequence could be in error as a result. Unless it is cancelled by another error in the ciphertext, an error in the decrypting sequence means an error in the plaintext.

One plaintext error is unavoidable for each ciphertext error, but with error propagation there will be extra plaintext errors. The ratio of the expected number of extra plaintext errors to the expected number of ciphertext errors will be called the increased noise  $N$ . The increased noise may depend on the key value, the  $f$  ciphertext bits that precede an erroneous ciphertext bit, the  $f$  ciphertext bits that follow it and the combinatorial circuit that computes the decrypting sequence. To examine the relationship between  $N$  and  $f$ , we will assume that in order to achieve pseudo-randomness in the encrypting sequence, a combinatorial circuit was chosen for which changing a non-empty subset of the bits in the feedback register has a fifty percent probability of changing the circuit's output. This probability is based on a uniform distribution of original values for the feedback register. A uniform distribution is expected when the encrypting sequence is pseudo-random. The probability is assumed to hold for all keys or at least for all keys that are ever used.

If there were no error propagation,  $N$  would be zero. If all ciphertext errors were single bit errors and were separated by at least  $f$  correct bits,  $N$  would be  $f/2$ . If the ciphertext errors were not always so well separated, they could be thought of as bursts. We will consider two errors separated by fewer than  $f$  correct bits to be in the same error burst. Bursts are thus separated by at least  $f$  correct bits but contain no stretches of correct bits that long.

If all ciphertext errors were in bursts of at least two bits,  $N$  would be  $(f + 1)/2pb + (1 - 2p)/2p$  where  $b$  is the mean length of the bursts and  $p$  is the probability that a bit within a burst is in error. The first and last bits of a burst must be in error to define the burst's position and length, and we assume that the intervening bits each have the same independent probability of being in error. That probability is  $(bp - 2)/(b - 2)$  and can be expected to fall between zero and one half.

If single bit errors are mixed with the burst errors,  $N$  rises toward  $f/2$ .

The increased noise is roughly proportional to the length of the

feedback register if the assumptions about the combinatorial circuit are met or nearly met. If as is likely to be the case, keeping the ciphertext error rate low is expensive and low plaintext error rates are valuable, only very short feedback registers may be practical.

#### USING LESS FEEDBACK

Deciding on a length for the feedback register is a problematic issue in selecting a cipher feedback system for an application. If the length is too short relative to the key length, the security of the system may be lost. If it is too long relative to the expected error bursts in the channel carrying the ciphertext, the plaintext may be too noisy to be useful or the cost of improving the ciphertext channel may be exorbitant.

One way to deal with the dilemma is to postpone it. It is better to choose a length for the feedback register for each message to suit the message's sensitivity and the condition of the ciphertext channel when the message is sent than it is to choose the length based on the worst case messages.

Some messages require the full measure of security that the key length provides. They should be sent when the channel noise is relatively quiet so that one can use a feedback register long enough not to impair security and still not cause an intolerable error rate in the plaintext produced by the decipherer.

Other messages may require some cryptographic protection but not as much. Not all secret messages need absolute protection. A message whose secrecy can be priced is protected for all practical purposes if the cost of thwarting the protection exceeds the value of the message's secrecy. Such messages can be sent even when the channel noise would be too bad for the long register by using only some of the register's length.

If only part of the feedback register is used as input to the stateless function that computes the encrypting and decrypting sequences of a feedback cipher system, the size of the codebook depends on the length used. For example, if the feedback register has ten bits of which only five are used to compute the encrypting and decrypting sequences, the codebook has thirty-two bits not a kilobit. The cost of breaking the system would be the cost of trying  $2^{32}$  codebooks assuming the key has at least thirty-two bits and the system has no other weaknesses. For some messages in some applications, this may be adequate

protection.

The effect on the noise of only using part of the feedback register depends on which part is used and on the nature of the errors in the ciphertext. If nearly all errors in the ciphertext are single bit errors, the increased noise in the plaintext depends on the length of the part used. For the example above in which only five bits of the feedback register are used, the increased noise would be the same as for a five-bit register all of which was used.

On the other hand, with bursty ciphertext errors, the increased noise depends primarily on the position of the bit used which is furthest from the input to the feedback register. For the example above in which five out of ten bits are used, the feedback register's bits are numbered from one to ten starting with the bit nearest the input. If the five bits used are the even-numbered bits, the increased noise would be as bad as if all ten bits were used. The noise would be just as bad if bits six through ten were used. But if bits one through five are the ones used, the increased noise is the same as if the feedback register only had five bits. In general, the greatest benefit in noise control comes from preferring to use the bits closest to the input.

#### A SOLUTION

Figure 2 shows the encipherer of a cipher feedback system allowing a dynamic choice of the effective feedback length. It has an  $f$ -bit cipher feedback register, an  $n$ -bit noise control register, a  $d$ -bit key register, an  $s$ -bit selection register, and  $n$   $s$ -bit circulating selector registers. All the registers are shift registers except possibly the noise control register and the  $d$ -bit key register.

The encipherer has a  $2^n$ -to-1 multiplexer with data inputs from the feedback register and address inputs from  $n$  modulo-2 multipliers. The multipliers each take an input from the noise control register and an input from one of the selector registers. The output from the multiplexer is input to the selection register.

The encipherer also has a stateless function with inputs from the selection register and the  $d$ -bit key register. The function's output is the encrypting sequence.

The contents of the  $n$  selector registers, the  $d$ -bit key register and the noise control register constitute the key. That makes the key length  $k$  be  $ns + d + n$ . Later discussion will show that  $k$  may actually be only  $ns + d$  since the noise control register is not like the other key registers.



The multiplexer's  $n$  address inputs and  $f$  data inputs make  $n$  be the base two logarithm of  $f$ , or the round-up of that log if  $f$  is not a power of two. For now we will assume that the length of the feedback register is a power of two.

The constant  $s$  should be at least the base two log of  $d$ . It is mostly constrained by the options for clock speeds.

There are two clocks called the feedback clock and the selection clock. The feedback clock shifts the feedback register. It pulses once for each ciphertext bit, or from the other points of view, once for each plaintext bit or once for each bit of the encrypting sequence.

The selection clock pulses  $s$  times faster than the feedback clock. It shifts the selector registers and the selection register. Thus for each bit of the encrypting sequence, the selector registers circulate exactly once and the selection register is filled with entirely new multiplexer outputs. This means that the encrypting sequence is a stateless function of the key and the feedback despite the existence of a selection register and the shifting of the selector registers. The noise control register and the  $d$ -bit key register remain fixed.

Each multiplexer output is called a feedback selection because it is a bit from the feedback. The products which are the outputs from the  $n$  multipliers determine which feedback bit is selected for each selection. The rightmost bit of the feedback register is selected when all the products are zeros. And when they are all ones, it is the leftmost bit. The other combinations of products select the other bits of the register according to the binary number formed by the products.

A zero in a bit of the noise control register leaves only half of the feedback bits able to be selected because the output from the multiplier receiving that zero will itself be zero regardless of the input from the selector register. In fact, a zero in the noise control register causes one of the selector registers to have no influence on any selections. Two zeros allow only one quarter of the feedback bits to be selected. Each additional zero eliminates half of the remaining choices. Having all zeros in the noise control register allows only the rightmost bit of the feedback register to be selected.

Because of the addressing order of the feedback bits, the bits that can be selected are the rightmost bits when the zeros in the noise control register are the bottommost bits. For example, if the noise control register has only two ones and they are the two topmost bits of the register, the  $s$  selections that go into the selection register will each be one of the four rightmost bits of the feedback register. Which of the four is selected in each case is determined by the corresponding bits in the two leftmost selector registers.

As we noted earlier, preferring the rightmost feedback bits, the ones nearest the input to the register, causes the least additional noise when ciphertext errors are bursty. (When errors are not bursty, preferences do not matter.) In the example, the noise would drop to the level expected of a feedback register with only four bits.

The decipherer for the cipher feedback system produces its decrypting sequence exactly as the encipherer produces the encrypting sequence. To recover the plaintext, the decipherer must be using the same key as the encipherer. The contents of the noise control registers must be the same, the contents of the d-bit key registers must be the same, and the contents of the corresponding selector registers must be the same when they are at the same point in their circulations. For each selector register, it is sufficient to know that the encipherer's register when its feedback clock pulses matches the decipherer's register when the decipherer's feedback clock pulses. It does not matter whether the two feedback clocks pulse at the same time.

#### OTHER FEEDBACK REGISTER LENGTHS

We had assumed that the length of the feedback register  $f$  was a power of two. The selection process is more complicated otherwise because some feedback bits can be selected by more than one address input to the multiplexer. This makes for irregularities in the effects of zeroes in the noise control register. As an example, we will consider a cipher feedback system in which  $f$  is 11. That makes  $n$  be 4. We will number the feedback bits from right to left as 1 through 11. The address input to the multiplexer will be referred to as the decimal equivalent of the binary number formed by the products with the topmost product being the most significant binary digit. The noise control value is similarly derived from the noise control register's contents with the topmost bit being most significant. The multiplexer selects bits from the feedback register according to the following table:

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Selection	1	2	3	3	4	5	5	4	6	7	7	8	9	9	10	11

The next table shows the number of feedback bits that can be selected for each noise control value. For example, a noise control value of eleven (1011) limits addresses to 0, 1, 2, 3, 8, 9, 10 and 11. This in turn allows selections to be made only from six feedback bits, bits 1, 2, 3, 6, 7, and 8.



Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Selectable	1	2	2	3	2	4	4	5	2	4	4	6	4	7	8	11

Provided that errors are not bursty, an appropriate choice of the noise control register's contents allows this feedback cipher system to be used with the same noise level as expected from other feedback cipher systems whose feedback registers had any number of bits up to eight. It could also be used with the noise level expected of its actual length of eleven.

Assuming that the key length is between 64 and 128, the noise control values of thirteen or more would not reduce the security of the system and the other noise control values would provide six different levels of security. Some of those six levels are probably too low for any messages in the application, but some choice still remains if not all messages require the security of having to try more than  $2^{64}$  codebooks or keys.

When errors are bursty, the noise level depends primarily on the distance from the input to the feedback register of the furthest bit that can be selected. The following table shows what that distance is for each noise control value. For example, the furthest bit that can be selected when the value is eleven is bit 8.

Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Distance	1	2	3	3	4	5	5	5	6	7	7	8	9	9	10	11

Because distances from one through eleven all appear in the table, the system can have the noise level expected of a cipher feedback system whose feedback register's length was any number up to eleven. Assuming again that the key length  $k$  is between 64 and 128, the choices that might be attractive are error control values of 0, 1, 3, 7, 11 and 13 which require trying 4, 16, 256,  $2^{32}$ ,  $2^{64}$ , and  $2^k$  codebooks or keys, respectively, and give effective feedback lengths of 1, 2, 3, 5, 8 and 9, respectively.

#### DYNAMIC NOISE CONTROL

The contents of the noise control register have been considered as part of the key. This is appropriate from some points of view but not from other points of view. If an attacker can measure the error rate, it is best not to consider the noise control value as part of the key. The error rates reveal too much about the noise control val-

ue to call it secret. On the other hand, the encipherer and decipherer must agree on the contents of their noise control registers just as with other registers that hold key material. If one changed the noise control value but not the rest of the key, the other would have to do the same.

There is another reason for the noise control value not to be secret like the key. It may be desirable to change the value much more often or more impulsively than the key is changed. If so, it would be convenient to communicate the new noise control values by an insecure channel.

Changing just the noise control register is not always safe. As an example of the danger, consider a cipher feedback system in which  $f$  is 8,  $n$  is 3,  $d$  is 30, and  $s$  is 10.  $k$  is 60, not including the noise control value in the key. Some messages are worth less than the cost of trying 64K codebooks, but others are worth more than the cost of trying  $2^{50}$  keys. If the cheaper messages were sent when there was one zero in the noise control register, a set of all possible values for the  $d$ -bit key register and two of the three selector registers could be found by trying  $2^{50}$  keys with the same value in the third selector register. The set is expected to have  $2^{34}$  members. This exposes the cheaper messages but at too high a price if they were the only target. Then when the zero is changed to a one and the key is not changed, the full value of the key can be found by trying  $2^{44}$  keys. A change of key would have been adequate to protect the more valuable messages, but changing just the noise control value was not enough.

Simple modifications to the registers holding key material can make it safe to change the noise control value without changing the key. One option is to eliminate the  $d$ -bit key register and use various positions from the  $n$  circulating selector registers instead for the inputs to the stateless function in Figure 2. The key length  $k$  would then be  $ns$ . Another option is to have only one circulating selector register and use  $n$  different positions for the inputs to the multipliers. This makes  $k$  be  $s + d$ . Combining those options reduces  $k$  to  $s$ .

Any of these options requires  $s$  or  $d$  to be larger to keep the same key length  $k$ . Hardware constraints are likely to put an upper limit on  $s$  and might sometimes limit  $k$  too much for some of these choices to be viable.

With any of the options, using a key with a short effective feedback length does not compromise the prior or subsequent use of the same key with a longer effective feedback length. Returning to the same example, while the cheaper messages were being sent with an effective feedback length of four bits, the codebook could be found by trying 64K

codebooks. This costs more than the cheaper messages are worth and is no help in exposing more valuable messages. Trying all  $2^{60}$  keys is expected to give a set of  $2^{44}$  possible keys, but trying fewer keys would give only an incomplete set of possible keys. After changing to an effective feedback length of eight for the more valuable messages, it would be worth searching the set of keys to expose those messages but the cost of producing the set exceeds the value of the more valuable messages.

Besides having two or more classes of messages with different values, having a choice of security levels and effective feedback lengths can be worthwhile if individual messages are not very valuable but the value of large aggregations of messages substantially exceeds the sum of their individual values. In an application with such messages, the noise control value can be adjusted to provide the maximum security at any particular time that does not produce intolerable noise in the plaintext. When the ciphertext channel is especially noisy, the security is too low for large aggregates but high enough for individual messages. Most of the time the ciphertext channel is not that noisy and the security level is high enough to protect the large aggregates of messages. Enough of the messages get the better protection that the valuable aggregates cannot be compiled at an affordable cost. The benefit is that messages do not need to be postponed when the ciphertext channel is especially noisy and that the ciphertext channel does not need to be as quiet all the time as the higher level of security and tolerance for plaintext noise would otherwise require.

#### CASCADED SYSTEMS

The encipherer in Figure 2 has an  $s$ -bit selection register and a stateless function with inputs from each of those  $s$  bits. The stateless function must be non-linear with respect to the selections if the whole system is to be non-linear. Also,  $s$  should be about as large as  $f$  or larger than  $f$ . A substantial portion of the cost of the system is the cost of the selection register and the stateless function.

Figure 3 shows an alternative to the selection register and stateless function of Figure 2. The original stateless function is replaced by a cascade of stateless functions. The total cost of the functions in each stage of the cascade is likely to be much less than the cost of the one large function. The key inputs to each stateless function come from the key registers.

Another savings comes from the replacement of the selection register with a much shorter cascade register for each stage in the cascade.

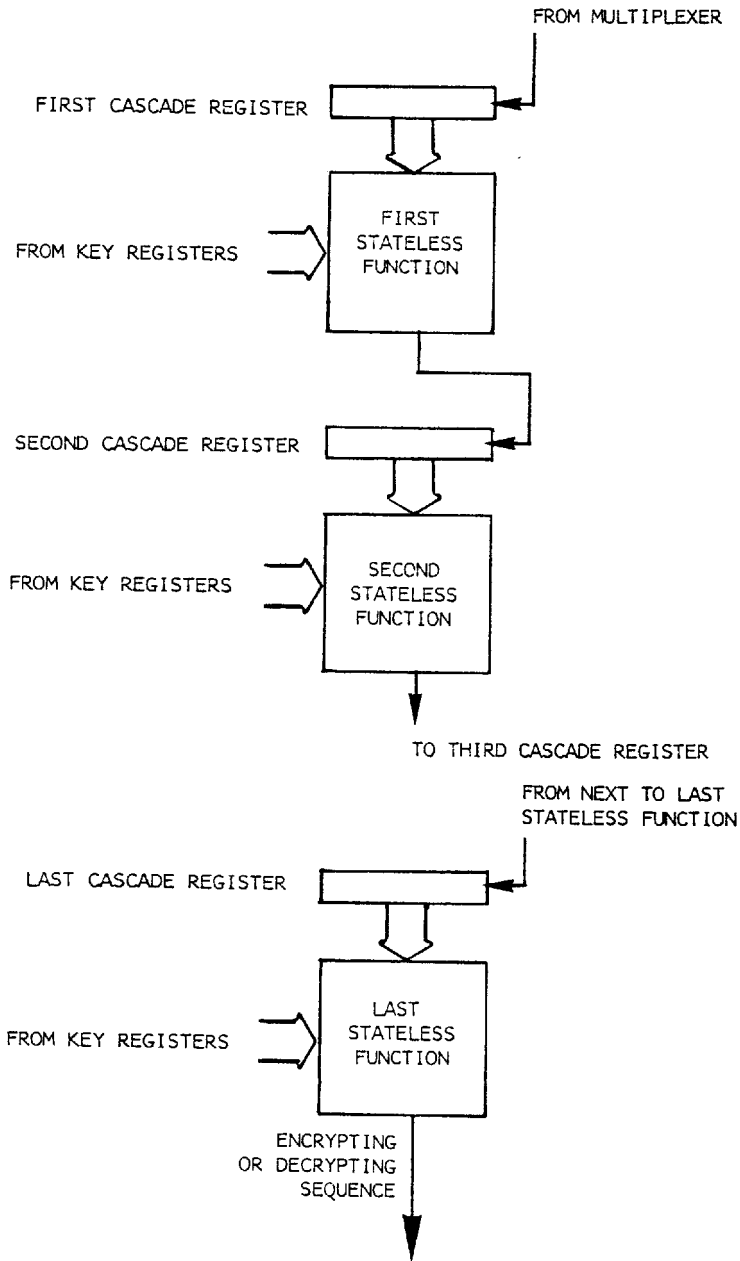
**CASCADE**

FIGURE 3

We will call the length of the cascade register for the  $i$ 'th stage  $c_i$ . The product of the lengths of the cascade registers is  $s$ , the length of the circulating selector registers.

The selections output from the multiplexer are input to the first cascade register. The outputs from each stateless function are input to the cascade register of the next stage in the cascade. The last stage is an exception. Its output is the encrypting (or decrypting) sequence.

Each stage in the cascade uses a different clock for shifting its cascade register. The first stage uses the selection clock, the clock which shifts the circulating selector registers. The clock for Stage  $i + 1$  pulses once after  $c_i$  pulses of the clock for the previous stage, stage  $i$ . This means that each output bit from a stage that is shifted into the next stage's cascade register is a function of a completely different selection of bits in its own cascade register.

Consider an example in which  $s$  is 36 and there are three stages.  $c_1$  is 4,  $c_2$  is 3 and  $c_3$  is also 3. The selection clock pulses 36 times faster than the feedback clock that shifts the feedback register. The clock for the second stage pulses 4 times slower than the selection clock which makes it 9 times faster than the feedback clock. And the clock for the third stage pulses 3 times slower than the clock for the second stage which makes it 3 times faster than the feedback clock. Each of the slower clocks can be cheaply derived from the next faster one.

Continuing with the example, each of the 36 selections from the feedback register for a bit of the encrypting sequence is shifted into the first stage's cascade register. After every fourth selection, an output from the first stage is shifted into the second stage's cascade register. We can partition the 36 selections for an encrypting bit into nine sets of four selections each with each input to the second stage depending on one of the nine sets. After every three of those sets, an output from the second stage is shifted into the third stage's cascade register. Another partition of the 36 selections into three sets of twelve each gives the selections on which each input to the third stage depends. When the feedback clock pulses, the output from the third stage which is a bit of the encrypting sequence, depends on three bits in the third stage's cascade register which depend in turn on all 36 selections.

The cascade of stateless functions remains a stateless function itself despite the cascade registers. This is because each cascade register is filled completely at least once between feedback clock

pulses and the inputs that are shifted into a cascade register while producing a bit of the encrypting or decrypting sequence depend only on feedback register bits selected since the last pulse of the feedback clock.

Returning to the example with three stages, it is easy to see what the equivalent stateless function would be in Figure 2. The selection register would have 36 bits. There would be nine copies of the first stage's function, each one taking its inputs from four bits of the selection register. There would be three copies of the second stage's function, each one taking its inputs from three of the nine first stage functions. Finally, there would be one third stage function whose inputs come from the second stage functions. Its output is the encrypting sequence.

## CONCLUSION

The cipher feedback systems presented provide away for some applications that need self-synchronization to suffer less from the trade-off between the security and error propagation problems common to all cipher feedback systems. Applications in which ciphertext noise varies or lower plaintext noise than the maximum tolerable has value can benefit if they have messages that vary in the degree of cryptographic security needed or if the value of an aggregation of messages exceeds the value of the individual messages.

The effective length of the cipher feedback register can be changed from one message to the next simply by agreeing on a new noise control value which need not be secret. Shorter effective lengths can be used for less valuable messages so that they will be corrupted with less noise.

The use of two different clocks allows the system to have a stateless function as necessary for self-synchronization. With even more clocks, a cascade can be used that helps to keep down the cost of the system.

# Efficient and Secure Pseudo-Random Number Generation.

(Extended Abstract)

*Umesh V. Vazirani\**

University of California

*Vijay V. Vazirani\*\**

Harvard University

Cornell University

\*Supported by NSF Grant MCS 82-04506, and by the IBM Doctoral Fellowship.

\*\*Supported by NSF Grant MCS 81-21431.

**Abstract:** Cryptographically secure pseudo-random number generators known so far suffer from the handicap of being inefficient; the most efficient ones can generate only one bit on each modular multiplication ( $n^2$  steps). Blum, Blum and Shub ask the open problem of outputting even two bits securely. We state a simple condition, the **XOR-Condition**, and show that **any generator** satisfying this condition can output  $\log n$  bits on each multiplication. We also show that the  $\log n$  least significant bits of RSA, Rabin's Scheme, and the  $x^2 \bmod N$  generator satisfy this condition. As a corollary, we prove that all boolean predicates of these bits are secure. Furthermore, we strengthen the security of the  $x^2 \bmod N$  generator, which being a Trapdoor Generator, has several applications, by proving it **as hard as Factoring**.

## 1. Introduction.

Recently, there has been a lot of interest in provably "good" pseudo-random number generators [10, 4, 14, 3]. These cryptographically secure generators are "good" in the sense that they pass all probabilistic polynomial time statistical tests. However, despite these nice properties, the secure generators known so far suffer from the handicap of being inefficient; the most efficient of these take  $n^2$  steps (one modular multiplication,  $n$  being the length of the seed) to generate one bit. Pseudo-random number generators that are currently used in practice output  $n$  bits per multiplication ( $n^2$  steps). An important open problem was to output even **two bits** on each multiplication in a cryptographically secure way. This problem was stated by Blum, Blum & Shub [3] in the context of their  $x^2 \bmod N$  generator. They further ask: how many bits can be output per multiplication, maintaining cryptographic security?

In this paper we state a simple condition, the **XOR-Condition** and show that any generator satisfying this condition can output  $\log n$  bits on each multiplication. We show that the XOR-Condition is satisfied by the  $\log n$  least significant bits of the  $x^2 \bmod N$  generator. The security of the  $x^2 \bmod N$  generator was based on Quadratic Residuosity [3]. This generator is an example of a *Trapdoor Generator* [13], and its trapdoor properties have been used in protocol design. We strengthen the security of this generator by proving it **as hard as factoring**. We also prove the XOR-Condition for  $\log n$  least significant bits of RSA/Rabin Schemes. Our proofs are based on recent developments in RSA/Rabin Scheme bit security. We present a history of these recent developments in the next paragraph. More recently, by a different proof Alexi, Chor, Goldreich & Schnorr [1] also proved the simultaneous security of  $\log n$  least significant bits of RSA/Rabin Schemes. Previously, Long & Wigderson [7] showed how to extract  $\log n$  bits at each stage from the generator of Blum and Micali [4]; however, this gain in efficiency is not enough to compensate for the extra time taken by this generator ( $O(n^3)$  steps for each stage).

The RSA-bit security problem has not only yielded several valuable proof techniques, but its two year history is also revealing in how mathematical progress is made - with successive partial solutions, simplifications and changes in point of view.

The first result on RSA bit security was proved by Goldwasser, Micali & Tong [6]. They proved that any oracle for RSA least significant bit (an efficient procedure which computes the least significant bit of the plaintext message when input the ciphertext) could be efficiently used to decrypt RSA messages, thus showing that RSA least significant bit is hard to compute unless RSA is easy to decrypt. However, the oracle was allowed to err on only  $\frac{1}{\log N}$  fraction of the inputs.

The next breakthrough came with the "binary gcd method" of Ben-Or, Chor & Shamir [2], which has been fundamental to all future developments. This procedure to decrypt RSA, probes the oracle at *pairs of points*, to determine the least significant bits of small messages. Each pair of probes is correct with probability  $1/2 + \epsilon$ , provided the oracle is correct on  $3/4 + \epsilon$  fraction of inputs, where  $\epsilon$  is any positive constant. They also showed that with more accurate oracles ( $7/8 + \epsilon$  correct) for other RSA bits they could



decrypt RSA.

At this stage it was not clear if even  $3/4$  security could be proved for the least significant bit. This question was resolved by Vazirani & Vazirani [12]. They showed that by *guessing* the least significant bits of  $\log\log N$  random small messages (which can be done in polynomial time by considering all  $\log\log N$  possibilities), they could randomize the oracle probes thereby decrypting with a less-than- $3/4$  oracle. They also give a method for extending the proof of security to  $\log\log N$  least significant bits and the xor's of all non-empty subsets of these bits. Goldreich [6] analyzed their combinatorial problem exactly and showed that less-than- $3/4$  could be interpreted as  $.725 + \epsilon$ .

In the next major development, Schnorr & Alexi used the strong Chernoff bound along with guessing least significant bits of  $\log\log N$  random messages to obtain a decryption procedure that used a *single oracle probe* for computing the least significant bit of small messages. Thus they proved  $1/2 + \epsilon$  security for any constant  $\epsilon$ . However, this security was still not good enough for using RSA for direct pseudo-random number generation -  $1/2 + 1/n^t$  security was needed.

By guessing  $\log\log N$  most significant bits of only two random numbers, Chor & Goldreich [5] showed how to generate  *$\log N$  pairwise independent numbers*, whose least significant bits were known. Thus they could ask the oracle  $\log N$  pairwise independent questions. Then using the Chebychev inequality, they show that a  $1/2 + 1/n^t$  oracle will suffice.

## 2. Extracting Two Bits from the $x^2 \bmod N$ Generator:

The  $x^2 \bmod N$  generator [3] is the following: On input  $N, x_0$  (where  $N$  is the product of two distinct primes each congruent to  $3 \bmod 4$ , and  $x_0$  is a quadratic residue mod  $N$ ), it outputs  $b_0 b_1 b_2 \dots$  where  $b_i = \text{parity}(x_i)$  and  $x_{i+1} = x_i^2 \bmod N$ . Its security was based on Quadratic Residuosity.

A variant of this generator outputs  $b_i = \text{location}(x_i)$ , where  $\text{location}(x) = 0$  if  $x < (N-1)/2$ ,  $1$  if  $x \geq (N-1)/2$ . The cryptographic security of this generator was also based on Quadratic Residuosity [3]. However, the generator which extracts parity as well as location at each stage may not be cryptographically secure, because revealing  $\text{parity}(x_i)$  may make  $\text{location}(x_i)$  predictable. Blum, Blum and Shub conjecture that this generator is also cryptographically secure, and ask the open problem: how many bits can be extracted at each stage, maintaining cryptographic security?

In this section we will prove their conjecture. In section 3 we will answer the open problem by giving a simple condition, the **XOR-Condition**. We will prove that  $\log n$  bits ( $n = |N|$ ) can be extracted at each stage from **any** generator satisfying this condition. We will also prove that the  $x^2 \bmod N$  generator as well as the generators based on RSA and Rabin's scheme satisfy this condition.

The following theorem will also give an intuitive idea for the general results of section 3, for which we will need to introduce some new definitions.

The 2-Bit  $x^2 \bmod N$  generator on input  $N, x_0$  ( $N$  and  $x_0$  as before), outputs  $a_0b_0a_1b_1 \dots$  where  $a_i = \text{parity}(x_i)$ , and  $b_i = \text{location}(x_i)$ ,  $x_{i+1} = x_i^2 \bmod N$ .

**Theorem 1:** The 2-Bit  $x^2 \bmod N$  generator is cryptographically secure.

**Proof:** Suppose the 2-Bit  $x^2 \bmod N$  generator is predictable to the left. There are two cases:

**Case 1:** It is predictable at an odd position, i.e. there is a probabilistic polynomial time procedure,  $P$ , which predicts  $b_{-1}$  with probability  $1/2 + \epsilon$ , given  $a_0b_0a_1b_1 \dots$ . Now we can use  $P$  to obtain  $\text{location}(x_{-1})$ : given any  $x_0$ , simply generate the sequence  $a_0b_0a_1b_1 \dots$ , and use  $P$  to obtain  $b_{-1} = \text{location}(x_{-1})$ . Contradiction, since  $\text{location}$  is secure under the Quadratic Residuosity Assumption [3].

**Case 2:** It is predictable at an even position, i.e. there is a probabilistic polynomial time procedure,  $P$ , which predicts  $a_{-1}$  with probability  $1/2 + \epsilon$ , given  $b_{-1}a_0b_0a_1b_1 \dots$ . Given  $x_0$ , we can generate  $a_0b_0a_1b_1$ , but not  $b_{-1}$ . Notice that  $P$  can be arbitrarily bad at predicting  $a_{-1}$  if it is not provided with the correct bit  $b_{-1}$ . So instead we will use  $P$  to obtain two procedures,  $P_1$  and  $P_2$ , such that either  $P_1$  has an  $\frac{\epsilon}{2}$ -advantage in guessing  $\text{parity}(x_{-1})$  or  $P_2$  has an  $\frac{\epsilon}{2}$ -advantage in guessing  $\text{parity}(x_{-1}) \text{ xor } \text{location}(x_{-1})$ .

Let  $u$  be the bit output by  $P$  on input  $0a_0b_0a_1b_1$ , and  $v$  be the bit output on  $1a_0b_0a_1b_1$ . If  $u = v$ ,  $P_1$  outputs  $u$ , else it outputs the flip of a fair coin. On the other hand,  $P_2$  outputs the flip of a fair coin if  $u = v$ , else it outputs  $u$  (in this case,  $u = (0 \text{ xor } u) = (1 \text{ xor } v)$ ). Notice the following facts:

- 1). On each input,  $x_0$ , exactly one of the two procedures,  $P_1$  and  $P_2$  uses the output of  $P$ , and the other one flips a coin.
- 2). Whenever  $P$  gives the correct answer, so does the procedure using its output; the other procedure, of course, flips a coin.

So the total number of correct answers output by both procedures is the number of correct answers output by  $P$  plus the number of correct answers output by the coin-flips. The fraction of total correct answers is  $(1/2 + \epsilon) + 1/2 = (1 + \epsilon)$ . So at least one of the two procedures must be correct on  $1/2 + \epsilon/2$  fraction of inputs. In Theorem 3, we will show that  $\text{parity}(x_{-1}) \text{ xor } \text{location}(x_{-1})$  is also secure, thus contradicting the existence of  $P_1$  and  $P_2$  and therefore  $P$ .

---

$x_{-1}$  is the unique square root of  $x_0 \pmod N$ , which is a quadratic residue.  $a_{-1} = \text{parity}(x_{-1})$  and  $b_{-1} = \text{location}(x_{-1})$ .

### 3. The XOR-Condition & Relative Security of Bits.

The difficulty in outputting two bits  $b_1(x)$  and  $b_2(x)$  at each stage (and the corresponding core of the above proof) lies in showing that there is no procedure that has any advantage in outputting bit  $b_2(x)$ , even though it is given  $b_1(x)$  for free, i.e. in showing the **relative security** of  $b_2$ , given  $b_1$ . In general, in order to output  $k$  bits securely at each stage from a pseudo-random number generator, the main fact to be proved is that for all  $i < k$ , there is no procedure that outputs bit  $b_{i+1}(x)$  given bits  $b_1(x), \dots, b_i(x)$ . In this section we shall prove that the XOR-Condition suffices to prove the relative security of these bits.

Blum & Micali [4] give sufficient conditions for using a one-way function and a boolean predicate for cryptographically secure pseudo-random number generation. In the past the security of boolean predicates (bits) has been proved by assuming the intractability of the underlying one-way function (e.g. in proving the security of RSA least significant bits). There are several forms for these intractability assumptions (in terms of circuit complexity, or Turing machine complexity, etc.). To make our theorems cleaner and independent of the nature of the intractability assumption, we shall in fact *define* the boolean predicate to be secure if the problem of inverting the underlying one-way function can be reduced in probabilistic polynomial time to the problem of computing the boolean predicate with a non-trivial advantage. We will require the reduction to be done uniformly (i.e. by the same Turing machine, for all  $N$ ). As a result, any reasonable intractability assumption for the underlying one-way function will translate into a similar security for the boolean predicate. Since this reduction process is the only known technique for proving bit security, the proposed simplification does not sacrifice generality for all practical purposes.

First, we define formally the underlying one-way function:

let  $N$  be a set of positive integers, the *parameter values*, and for each  $N \in \mathbf{N}$ , let  $n = |N|$  and  $X_N \subset \{0,1\}^n$  be the *domain*. We will assume that a random element of  $X_N$  can be generated.

$E_N: X_N \rightarrow X_N$  is the one-way function with parameter  $N$ .

$b: (N, x) \rightarrow \{0,1\}$  is a *boolean predicate* computable in prob. poly. time.

**Definition:** Oracle  $O_{b,N}$  has an  $1/2 + \epsilon$  **advantage** in computing the boolean predicate  $b$ , if for  $1/2 + \epsilon$  fraction of domain elements  $x \in X_N$ ,  $O_{b,N}$  outputs  $b(x)$  on input  $E_N(x)$ .

**Definition:** Boolean predicates  $b_1, \dots, b_{k(n)}$  are **inversion secure** if for each  $t > 0$  there is a Las Vegas Algorithm  $T$  that runs in prob. poly. time:

$$T^{O_{b_i,N}}[i, E_N(x)] = x.$$

where  $O_{b_i,N}$  is a  $1/2 + 1/n^t$  advantage oracle for  $b_i$  with respect to  $N$ .

**Definition:** Oracle  $O_N$  has a  $1/2 + \epsilon$  advantage for boolean predicate  $b_l$  **relative to**  $b_1, \dots, b_{l-1}$  if for at least  $1/2 + \epsilon$  fraction of  $x \in X_N$ ,

$$O_N[E_N(x), b_1(x), \dots, b_{l-1}(x)] = b_l(x).$$

The behavior of  $O_N$  is unspecified, and may be arbitrarily bad, if any of the  $l-1$  bits is incorrectly input.

**Definition:**  $b_i$  is **secure relative to**  $b_1, \dots, b_{i-1}$  if for each  $t > 0$ , there is a Las Vegas algorithm  $T$  which runs in polynomial time:

$$T^{O_N}[E_N(x)] = x$$

where  $O_N$  is a  $1/2 + 1/n^t$  advantage oracle for  $b_i$  relative to  $b_1, \dots, b_{i-1}$ .

Notice that  $T$  can use the oracle effectively only if it can guess correctly each of  $b_1(x) \dots b_{i-1}(x)$ . But in our case, these boolean predicates are already inversion secure. For this reason, proving relative security (i.e. the existence of  $T$ ) is considerably more difficult than proving simple bit security. We now give the **XOR-Condition**, and show (in Theorem 2) how it yields a proof of relative bit security from simple bit security.

**The XOR-Condition:** Boolean predicates  $b_1 \dots b_k$  satisfy the XOR-Condition if the XOR of each on-empty subset of these predicates is inversion secure.

**Theorem:** Let  $k(n) = O(\log n)$ . Let  $b_1, \dots, b_{k(n)}$  be boolean predicates which satisfy the XOR-condition. Then for every  $i < k(n)$ ,  $b_i$  is secure relative to  $b_1, \dots, b_{i-1}$ .

**Proof:** Suppose that  $O_N$  is a  $1/2 + 1/n^t$  advantage oracle for  $b_i$  relative to  $b_1, \dots, b_{i-1}$ . Let  $T$  be the efficient procedure in the definition of the XOR-Condition for  $b_1, \dots, b_{k(n)}$ . Then  $T'$  is an efficient Las Vegas algorithm which uses the oracle  $O_N$  to invert  $E_N$  (see explanation at the end of the procedure).

$T'$ : On input  $N, i, E(x)$ ;  
 $O'_N \leftarrow \text{Construct-Oracle}[O_N, i-1, 1/n^t]$ ;  
 Run  $T$  with oracle  $O'_N$  to invert  $E_N$ .  
 end;

**Construct-Oracle:** On input  $O_N, j, \epsilon$ ;  
 If  $j = 0$  then return  $O_N$ .  
 Else, let  $u = O_N[E(x), b_1, \dots, b_{j-1}, 1]$   
 $v = O_N[E(x), b_1, \dots, b_{j-1}, 0]$ ;  
 Oracle  $O_1[E(x), b_1, \dots, b_{j-1}] =$   
 $\begin{cases} u & \text{if } u=v, \\ \text{the flip of a fair coin} & \text{otherwise} \end{cases}$   
 Oracle  $O_2[E(x), b_1, \dots, b_{j-1}] =$   
 $\begin{cases} v & \text{if } u \neq v, \\ \text{the flip of a fair coin} & \text{otherwise} \end{cases}$

Sample the two oracles on  $8\epsilon^2 \log n$  random elements of  $X_N$ , to determine the fraction of correct answers given by each.

If  $O_1$  gives atleast  $1/2 + \epsilon/4$  fraction correct answers then  
 return(Construct-Oracle[ $O_1, j-1, \epsilon/4$ ]).  
 Else return(Construct-Oracle[ $O_2, j-1, \epsilon/4$ ]).  
 end.

$T'$  first calls the recursive procedure Construct-Oracle. By a proof similar to that of Theorem 1 either there is an oracle having  $1/2 + 1/2n^t$ -advantage for  $b_i$  relative to  $b_1 \cdots b_{i-2}$  or there is an oracle having  $1/2 + 1/2n^t$ -advantage for  $b_{i-1} \text{ XOR } b_i$  relative to  $b_1 \cdots b_{i-2}$ . Moreover, sampling the two oracles a polynomial ( $8n^{2t} \log n$ ) number of times, Construct-Oracle can determine with high probability ( $1 - 1/2 \log n$ ) which of the two oracles has the advantage. Continuing this procedure  $i-1$  times, Construct-Oracle will obtain, with probability  $\geq 1/2$ , a  $1/2 + 1/n^{t+\epsilon}$ -advantage oracle for the XOR of some subset of  $b_1 \cdots b_i$ .  $T'$  can now use this oracle to invert  $E_N$ .

#### 4. Proving the XOR-Condition, and Improving the Security of the $x^2$ -mod N Generator.

In this section we state the theorems on the XOR-Condition, and briefly sketch the main ideas of their proofs. Detailed proofs will follow in the final paper.

**Theorem 3:** The *parity* function of the  $x^2$ -mod N generator is as hard as factoring, i.e. for any  $t > 0$ , an oracle which has a  $1/2 + n^t$  advantage in guessing *parity*( $x$ ) on input  $x^2 \text{ mod } N$  can be used to factor N. Here  $x$  is the unique square root of  $x^2 \text{ mod } N$  which is a quadratic residue.

By modifying the algorithm of [1], we show how to use the *parity* oracle to extract square roots mod N efficiently. The main difficulty is that the parity oracle gives the least significant bit of the square root which is a quadratic residue. Thus on query  $x^2 \text{ mod } N$ , if  $x$  is a quadratic residue mod N, the oracle will give  $\text{lsb}(x)$ . Else, it gives the complement. In some sense the oracle gives the  $\text{lsb}(x)$  encrypted within the hard function - quadratic residuosity. How can the oracle's answers be interpreted correctly? The key idea is that a parity oracle with a small advantage can be used to implement a residuosity oracle which is correct with overwhelming probability [3]. This residuosity oracle may be used to "decrypt" the answers of the parity oracle.

This proves that the *location* function is also as hard as factoring since a  $1/2 + 1/n^t$  oracle for *location* can be converted to a  $1/2 + 1/n^t$  oracle for *parity*:  
 $\text{parity}(x) = 0$  iff  $\text{location}(x/2) = 0$ .

**Theorem 4:** The function *parity xor location* of the  $x^2$ -mod N generator is as hard as factoring.

We simply note that in the decryption algorithm [1], we already know the *location* of the numbers we query about. So the oracle for *parity xor location* is in effect giving us *parity*, which is sufficient for decryption.

**Theorem 5:** For each non-empty subset  $S$ , of the  $\log\log N$  least significant bits, of the  $x^2 \bmod N$  generator: obtaining a  $1/2 + 1/n^t$  advantage in guessing the XOR of these bits is as hard as factoring.

The idea presented in [12] will suffice along with the decryption algorithm [1]. Let the most significant bit being considered in  $S$  be the  $k^{\text{th}}$  bit,  $k \leq \log\log N$ . Instead of running the gcd algorithm on "small messages" in the interval  $[-\epsilon N, \epsilon N]$ , we will choose the small messages in the interval  $[-\epsilon N/2^{k-1}, \epsilon N/2^{k-1}]$ . Now, for any  $x$  in this smaller interval,  $\text{lsb}(x) = 1$  iff  $\text{XOR}_{S'} 2^{k-1}x = 1$  because the  $k-1$  least significant bits of  $2^{k-1}$  are all 0's. So, in order to obtain the lsb of a number in the interval  $[-\epsilon N/2^{k-1}, \epsilon N/2^{k-1}]$ , we simply multiply it by  $2^{k-1}$ , and run the modified algorithm of Theorem 3. In a similar manner, we can prove the XOR-Condition for RSA and Rabin Scheme also:

**Theorem 6:** For each non-empty subset  $S$ , of the  $\log\log N$  least significant bits, of RSA/Rabin Schemes: obtaining a  $1/2 + 1/n^t$  advantage in guessing the XOR of these bits is as hard as decrypting RSA/factoring.

## 5. Going Beyond $\log n$ Bits.

How many bits can we hope to extract securely on each multiplication? We first make two simple observations. Certainly not all  $n$  bits. Because then all boolean predicates of  $x$  will be secure even though  $E(x)$  is given. But, for example for RSA, we know that  $\text{Jacobi Symbol}(x) = \text{Jacobi Symbol}(E(x))$ , which is efficiently computable. Secondly, notice that in all the proofs,  $\log n$  can be replaced by  $c \log n$ , for any constant  $c$ .

In proving bit security, we limited the reductions (algorithms to decrypt the one-way function, using oracle for the bit) to be probabilistic polynomial time. If the computational complexity of the underlying one-way function is much more than a polynomial, then there is no reason to put this restriction. For example, if the intractability assumption on the underlying one-way function states that its computational complexity is  $o(n^{\log n})$ , then the reduction can be allowed  $O(n^{\log n})$  time. In this case our proofs can be modified to show that  $\log^2 n$  least significant bits satisfy the XOR-Condition with  $1/2 + 1/n^{\log n}$  security for the bits and their XORs. These  $\log^2 n$  bits can be output by a pseudo-random number generator, by a simple modification of the proof of Theorem 2. In general, if the the assumption on the complexity of the underlying one-way function is  $o(f(n))$ , then our proofs extend to showing that  $\log(f(n))$  bits can be securely output at each stage. For example, presently the fastest factoring algorithm runs in time

$O(2^{\sqrt{n \log n}})$  []. So, if we assume  $f(n) = 2^{\sqrt{n}}$ , we can securely extract  $\sqrt{n}$  bits on each multiplication.

**6. Acknowledgements:** We are extremely grateful to Lenore Blum, Manuel Blum, Michael Rabin and Les Valiant for some very fruitful discussions.

## 7. References.

- 1). W. Alexi, B. Chor, O. Goldreich & C. Schnorr, "RSA/Rabin Bits are  $1/2 + 1/\text{poly}(\log N)$  Secure," this conference.
- 2). M. Ben-Or, B. Chor and A. Shamir, "On the Cryptographic Security of RSA bits," 1983 STOC.
- 3). L. Blum, M. Blum and M. Shub, "A Simple Secure Pseudo-Random Number Generator," to appear in SIAM Journal of Computing.
- 4). M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," 1982 FOCS.
- 5). B. Chor and O. Goldreich, " , " in preparation.
- 6). O. Goldreich, "On the number of Close-and-Equal Pairs of Bits in a String (with implications on the security of RSA's L.s.b.)", MIT/LCS/TM-256, March 1984.
- 7). S. Goldwasser, S. Micali and P. Tong, "Why and How to Establish a Private Code on a Public Network," 1982 FOCS.
- 8). D. Long and A. Wigderson, "How Discreet is the Discrete Log?" 1983 STOC.
- 9). M. O. Rabin, "Digital Signatures and Public-key Functions as Intractable as Factorization," MIT/LCS/TR-212 Tech. memo, MIT, 1979.
- 10). C. Schnorr and W. Alexi, "RSA-bits are  $0.5 + \epsilon$  secure," 1984 EURO-CRYPT.
- 11). A. Shamir, "On the Generation of Cryptographically Strong Pseudo-Random Sequences," 1981 ICALP.

- 12). U. Vazirani and V. Vazirani, "RSA bits are  $.732 + \epsilon$  secure," CRYPTO-83.
- 13). U. Vazirani and V. Vazirani, "Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design," 1983 FOCS.
- 14). A. Yao, "Theory and Applications of Trapdoor Functions," 1982 FOCS.



## AN LSI RANDOM NUMBER GENERATOR (RNG)

R. C. Fairfield, R. L. Mortenson, & K. B. Coulthart

AT&T Bell Laboratories  
25 Lindsley Drive  
Morristown, New Jersey 07960

### INTRODUCTION

This paper describes a CMOS digital LSI device which generates a random bit stream based on the frequency instability of a free running oscillator. The output of the device is a truly random binary number; not pseudo random.

The device was developed to be used, principally, in cryptographic systems as a source for cryptographic keys and/or initial values. Some cryptographic systems rely on pseudo random generator schemes as a source of keys and initial values but a cryptographically secure system demands the use of truly random numbers.

### DESIGN OBJECTIVES

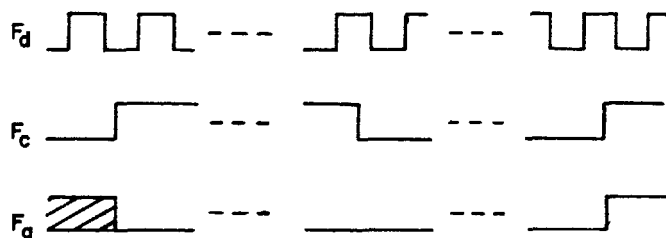
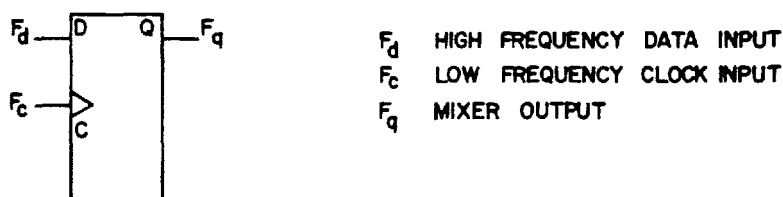
At the outset of the device development four design objectives were established.

- (1) The fundamental source of the random binary number had to be based on a natural statistical or probabilistic phenomenon.
- (2) A completely digital IC was desired.
- (3) The device should operate in a microprocessor controlled environment. That is, a microprocessor compatible interface had to be provided.
- (4) Means to run periodic checks of the circuitry had to be provided.

## IMPLEMENTATION

The fundamental probabilistic phenomena utilized on the device is the frequency instability of a free running oscillator. The use of this phenomena to generate truly random numbers is not new. The RAND Corporation used this phenomena to generate a table of a million random digits which it published in 1955 [1].

In the device under discussion here, a random bit stream is generated by digitally mixing two independent square waves in a positive-edge-triggered D-type flip-flop. In the implementation, a low frequency wave is used to clock the flip-flop and in so doing sample a high frequency wave which is applied to the flip-flop input data lead. The circuit is shown in Figure 1.



DIGITAL MIXER

Figure 1

The IC contains two on-chip oscillators. One produces a high frequency nonadjustable 8 MHz square wave which can be used as the sampled signal at the data input. The second oscillator frequency is adjustable with external resistors and capacitors and can be used as the sampling signal applied to the clock lead of the mixer. The data

and clock signals can also be supplied from off chip sources. It is not recommended that the 8 MHz on-chip oscillator be used. Weak coupling or interaction between the two oscillators has been observed and this may affect the randomness of the number produced. The use of an off chip high frequency square wave oscillator, preferably crystal controlled, will substantially reduce any possibility of coupling. The use of an RC controlled oscillator for the sampling signal is favored over an LC or crystal controlled oscillator in this application because of the inherently lower Q of the frequency determining circuits. The result is higher noise and poorer spectral purity and thus larger period variations in the output waveform [2], [3].

If the high frequency signal has a 50% duty cycle and the low frequency clock has significant cycle to cycle period variation, each successively generated bit is independent and has equal probability of being a "one" or a "zero". What is meant by significant is defined below. Of course, neither of these conditions hold in general. Thus, deterministic circuitry must be used to eliminate bias caused by less than ideal signals.

If the high frequency sampled square wave has something other than a 50% duty cycle there will be a bias toward either "one" or "zero" bits at the sampling D-type flip-flop output. This bias can be effectively removed if groups of samples are passed through an exclusive-or chain. Figure 2 shows an implementation which generates a single random bit by taking the exclusive-or sum of four stored bits. The data rate is reduced by four since bits are not reused. This circuit is implemented on the IC with the left most or first flip-flop performing the digital mixing of the two applied square waves. The bias correction achieved by such a circuit is given by the following. If the duty cycle of the high frequency square wave is  $p$ , the probability of obtaining a "one" as a sample is  $p$  while the probability of obtaining a "zero" is  $1-p$ . However, if  $n$  independent samples obtained from a biased signal are passed through an exclusive-or chain, the probability that the bit at the output of the chain is "one" is given by

$$P_X(1) = 0.5 - 2^{n-1}(p-0.5)^n$$

while the probability that the bit is a "zero" is given by

$$P_X(0) = 0.5 + 2^{n-1}(p-0.5)^n$$

Thus if the duty cycle of the high frequency square wave is 55% and the exclusive-or operation is performed over four samples, the probability of obtaining a "one" out of the exclusive-or chain is

$$P_x(1) = 0.49995$$

while the probability of obtaining a "zero" is

$$P_x(0) = 0.50005$$

In the above equations it can be seen that as  $n$  goes to infinity the probability approaches 0.5.

## PARITY FILTER

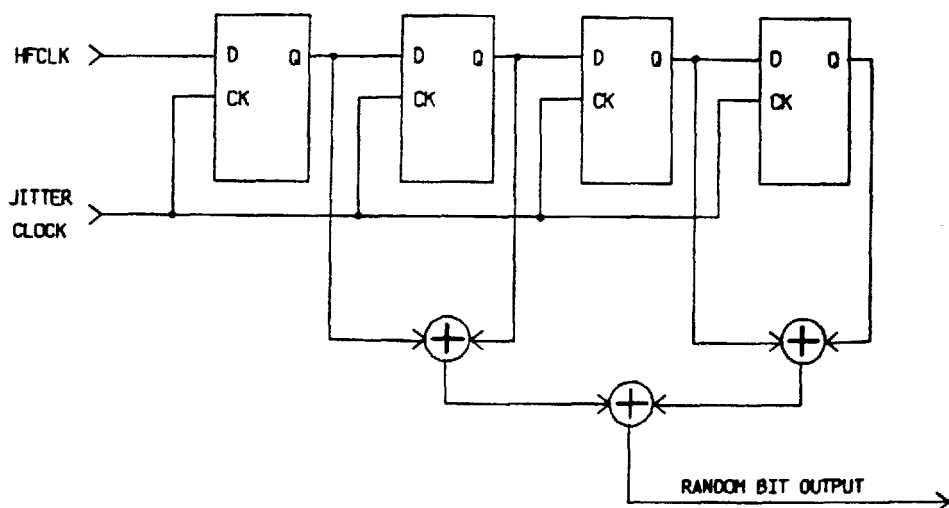


Figure 2

The second bias or difficulty which arises stems from insufficient phase jitter or frequency fluctuations on the clock input. As previously stated, if the low frequency clock has significant cycle to cycle period variation, each successively generated sample is independent and an individual cannot accurately predict the state of a sample knowing the state of the preceding sample and the mean frequencies of the two signals generating them. Figure 3 and Table 1 show the probability of guessing a bit in sequence knowing the expected periods of the high and low frequency oscillators, the standard deviation of the low frequency oscillator's

period variations, plus the state of the previous bit.

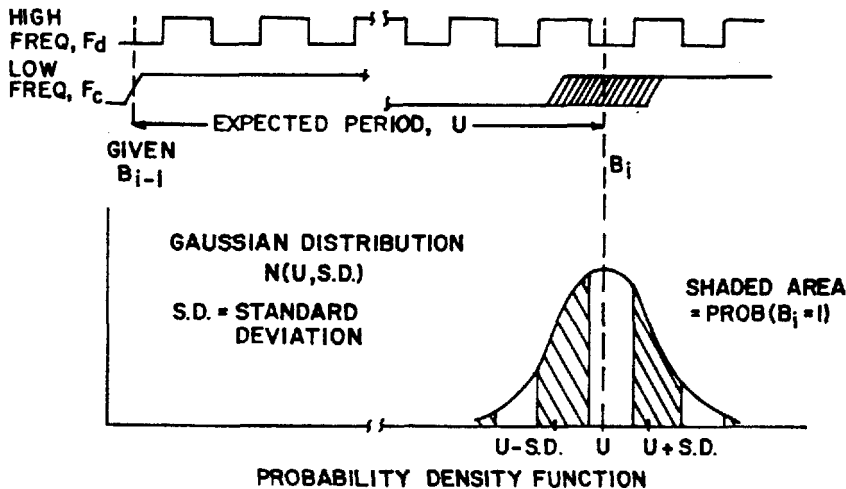


Figure 3

2 x S.D./Td *****	PROB[B <sub>i</sub>  B <sub>i-1</sub> ] *****
2.00	0.50000+
1.50	0.500006
1.25	0.500180
1.00	0.502891
0.75	0.525041
0.50	0.617052
0.25	0.797871
0.20	0.837035
0.10	0.913442

S.D. = standard deviation of the low frequency period variation.

Td = period of the high frequency oscillator.

Table 1

It is clear from Table 1 that if twice the standard deviation of the low frequency period variation is but a fraction of the high frequency oscillator period there is significant bit to bit correlation and individual bits can be guessed from the state of preceding bits. On the other hand, if the ratio of twice the standard deviation of the

low frequency period variation to the high frequency oscillator period is greater than 1.5 there is little bit to bit correlation.

In general one does not get cycle to cycle period variations from the D-type flip-flop clock such that the variations span 1.5 or more cycles of the data input signal. Thus, there will be sample to sample correlation between bits out of the flip-flop and knowing one sample and the mean frequencies of the input signals one can predict the state of the next sample with some degree of accuracy. As with the duty cycle bias the sample to sample correlation may be effectively removed through the use of exclusive-or circuits. The correlation correction is achieved as samples generated many clock cycles apart are exclusive-ored together. The circuit in Figure 4, which appears on the IC and is fed from the output of the parity filter shown in Figure 2, performs this task.

### SCRAMBLER CIRCUIT

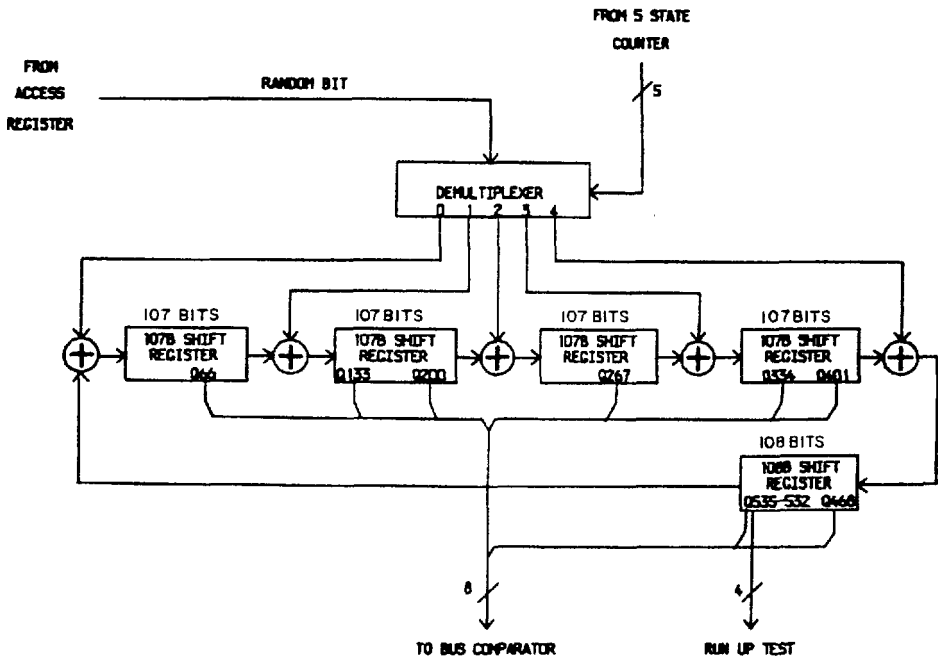


Figure 4

The magnitude of the correlation correction stems from the Gaussian distribution model which can be used for the low frequency oscillator period variations. The Gaussian distribution has the property that a

change in the random variable (the period) yields a like change in the standard deviation. This linear property has been experimentally verified in the case of the on-chip oscillator. If we consider samples taken ten cycles apart, as opposed to successive samples, the standard deviation of the tenth clock edge with respect to the first clock edge is tenfold the standard deviation between successive clock edges. Thus, samples taken many cycles apart have low correlation and the lower the correlation of the samples passing through an exclusive-or network, the lower the probability of predicting the state of the exclusive-or output with any degree of certainty. The spacing of the first five register samples which are exclusive-ored in the circuit of Figure 4 is shown in Table 2.

# SCRAMBLER REGISTER CONTENT AFTER 2680 SAMPLES

\*\*\*\*\*

REGISTER NUMBER

CONTENT

\*\*\*\*\*

\*\*\*\*\*

1	307+736+1700+2129+2558
2	413+842+1271+2235+2664
3	90+519+948+1377+1806
4	625+1054+1483+1912+2341
5	196+1160+1589+2018+2447
6	302+731+1695+2124+2553
7	408+837+1266+2230+2659
8	85+514+943+1372+1801
9	620+1049+1478+1907+2336
10	191+1155+1584+2013+2442
11	297+726+1690+2119+2548
12	403+832+1261+2225+2654
13	80+509+938+1367+1796
14	615+1044+1473+1902+2331
15	186+1150+1579+2008+2437
16	292+721+1685+2114+2543
-	-
107	308+737+1166+2130+2559
-	-

The symbol + stands for an exclusive-or operation.

Table 2

The table should be read as follows: After 2680 samples have been input to the scrambler, the content of scrambler register one is the exclusive-or sum of the 307th, 736th, 1700th, 2129th, & 2558th samples. As is evident from Table 2, the registers in the scrambler contain the exclusive-or sum of samples generated 429 or 964 clock cycles apart or the exclusive-or sum of independent samples. Also upon examination of registers 1 and 107 it is apparent that pairs of successively generated bits do tend to accumulate in pairs of registers. However, this pair accumulation is not complete in that one pair of bits in registers 1 and 107 are samples which are generated 534 clock cycles apart. Samples generated 534 cycles apart are independent. Looking down the register contents we see that the closest correlation exists between registers spaced five apart since they contain pairs of samples all of which are generated five clock cycles apart.

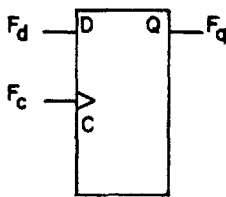
One item to note is that as signals feed into the scrambler and are exclusive-ored with other samples, the duty cycle bias correction is enhanced. Actually, the parity filter shown in Figure 2 is not necessary and if it were removed from the IC, the generation and accumulation of a random number in the scrambler circuit would occur four times faster.

## EXPERIMENTAL RESULTS

The effects of frequency instabilities on the digital mixing operation is best illustrated through the digital mixer transfer function shown in Figure 5. At the output of the mixer it is most appropriate to speak of output sequency as opposed to the output frequency. One can speak of the mixer output frequency only in the sense of the fixed number of transitions which occur in a given period for, in general, the transitions are not evenly spaced in time. Thus, we speak of output sequency or the output patterns generated. As evident from Figure 5, the transfer function of the digital mixer is a periodic function with period equal to the frequency of the clock input. The output sequency varies from: a steady string of "ones" or "zeros" when the data lead frequency is an integral multiple of the clock; to the most rapidly varying sequence of alternating "ones" and "zeroes" which occurs when twice the data lead frequency is an odd integral multiple of the clock frequency. At other data lead



frequencies, different patterns are generated.



$F_d$  HIGH FREQUENCY DATA INPUT  
 $F_c$  LOW FREQUENCY CLOCK INPUT  
 $F_q$  MIXER OUTPUT

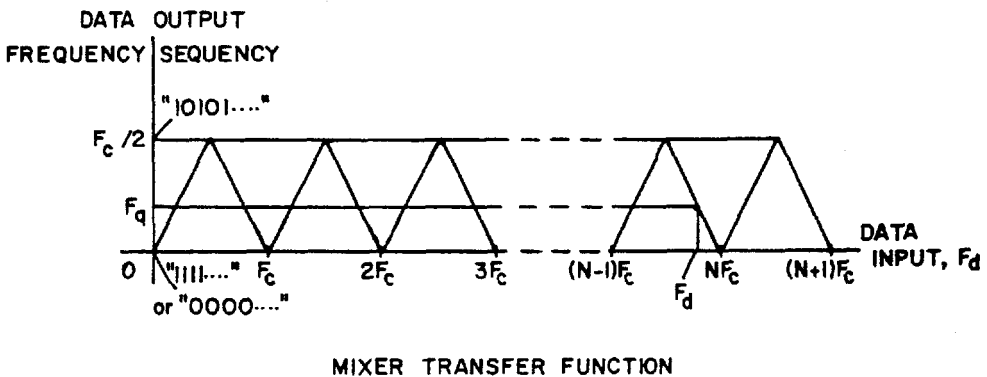


Figure 5

In the presence of clock lead frequency or period fluctuations, the digital mixer output sequency fluctuates. In the current application one would like the frequency fluctuations to be large enough to cover all possible output patterns. Of course, this in general will not occur. To determine what will occur one must know the extent of the fluctuations of the oscillators being used. Two types of parameters are generally of interest in describing oscillators: the short term or instantaneous frequency variations and the long term frequency drift. Figure 6 shows the experimentally measured (short term) fluctuations of the on-chip RC oscillator at two different frequencies. Figure 7 shows the effects of the measured low frequency period variations on the output sequency given a data lead frequency in the vicinity of 8.1 MHz. In Figure 7, the mixer transfer characteristics for the experimentally measured clock oscillator frequencies are plotted about the high frequency data lead operating point. The solid lines represent the mixer characteristic with the clock frequency set to its nominal value minus the standard deviation and the dashed lines

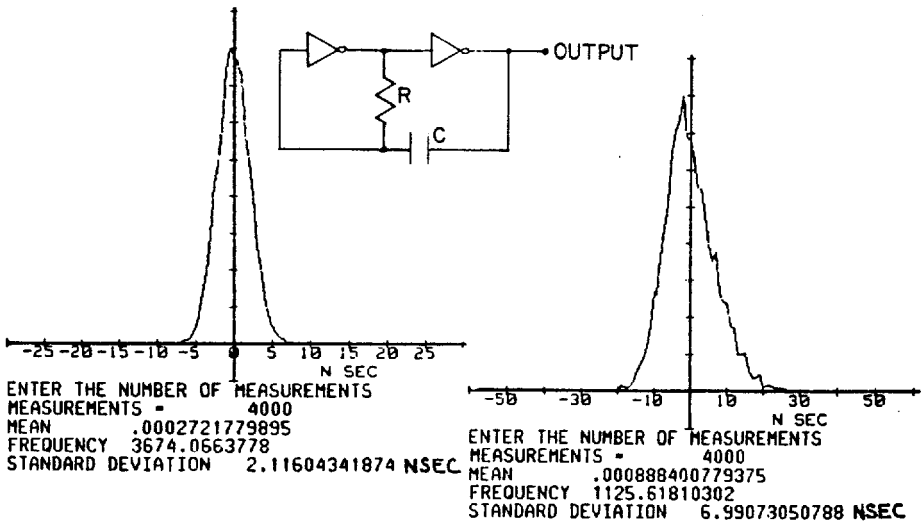


FIGURE 6 LOW FREQUENCY OSCILLATOR CHARACTERISTICS

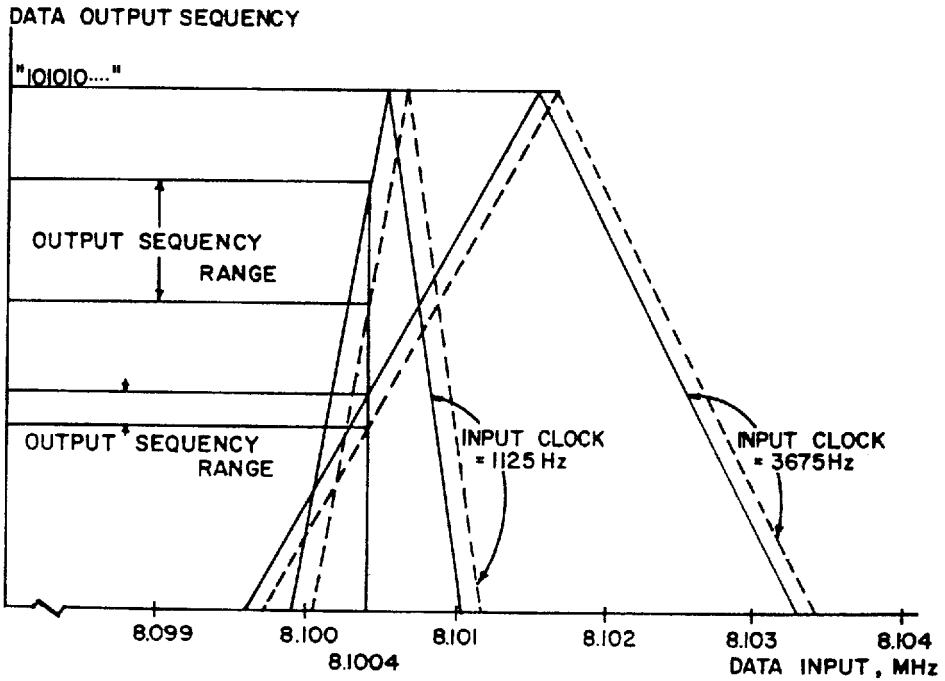


FIGURE 7 MIXER RESPONSE TO FREQUENCY INSTABILITY

represent the characteristic for the clock set to its nominal value plus the standard deviation. For a completely stable high frequency, only a portion of the possible output patterns are covered by the frequency variations. For the 3.674 KHz signal, approximately 7% of the possible patterns will be generated while for the 1.125 KHz signal, approximately 23% of the patterns will be generated. This, of course, indicates that the lower clock frequency is more desirable for the generation of the random number. Figure 8 shows the long term drift of the on-chip RC oscillator mean frequency when operating about 1.125 KHz.

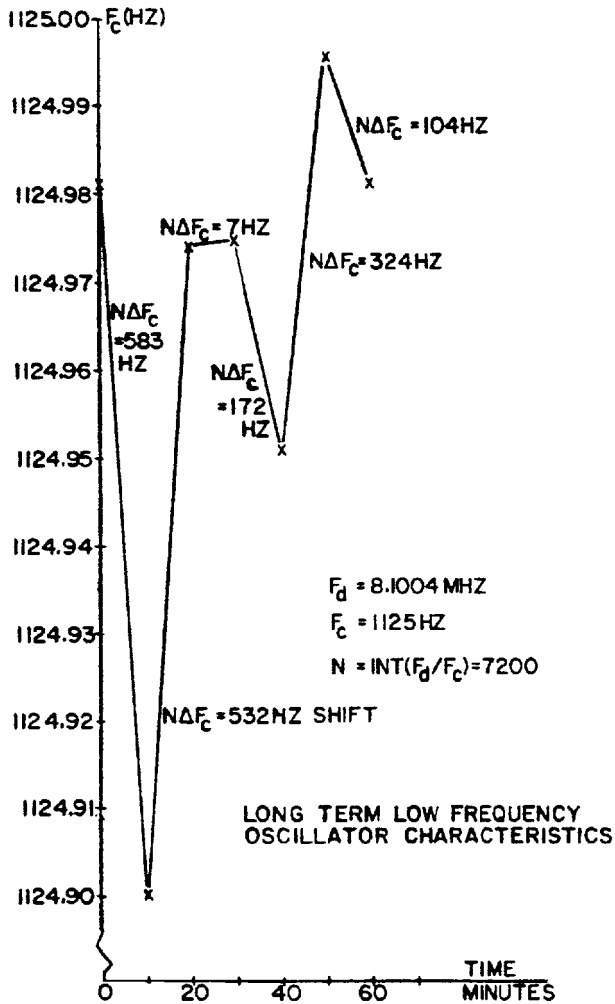


Figure 8

This variation translates to an almost 700 Hz shift in the transfer function about the 8.1 MHz clock. Thus, over a long period of time (hours) one can expect the operating point to be uniformly distributed over all possible patterns.

Figure 9 shows the power spectrum of the data coming out of the IC's D-type flip-flop mixer.

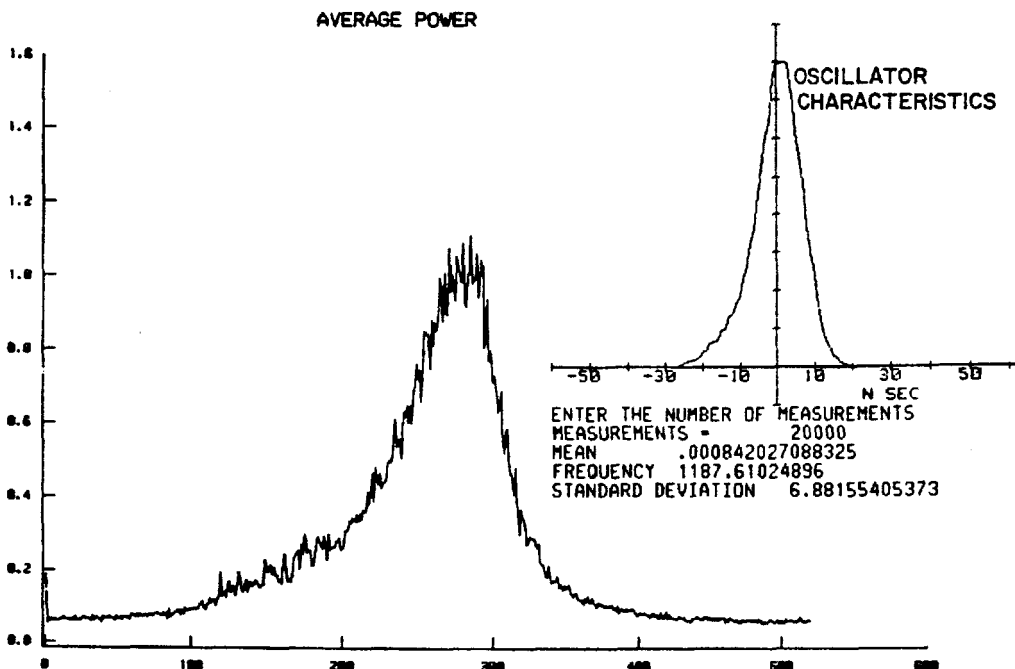


Figure 9

Data was broken up into blocks of 1024 nonoverlapping samples and the power spectrum was estimated and averaged over a file of approximately a quarter of a million points. One sees that the power is confined to a band of frequencies (sequences) about some mean. Figure 10 shows the power spectrum after exclusive-oring adjacent samples and scrambling groups of five bits as the scrambler circuit of Figure 4 would. One sees that after scrambling the energy is uniformly spread over all frequencies as it should be for a truly random bit stream.

Figure 11 shows the power spectrum of a second set of D-type flip-flop mixer samples. This time the spectrum is centered about a different mean frequency and the deviation from the mean is smaller. Also, one sees a d.c. term indicating a local bias (within 1024

## AVERAGE POWER

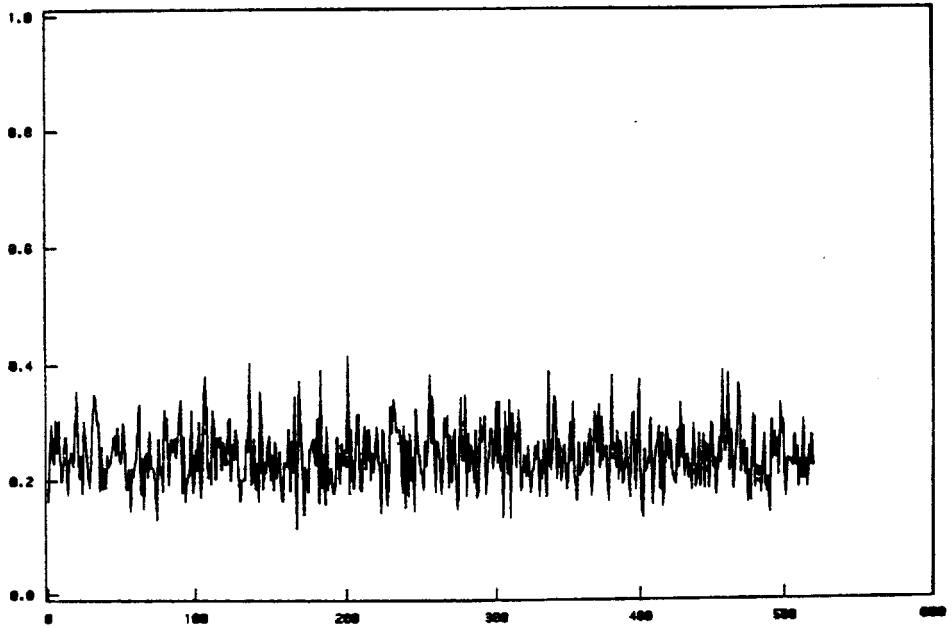


FIGURE 10

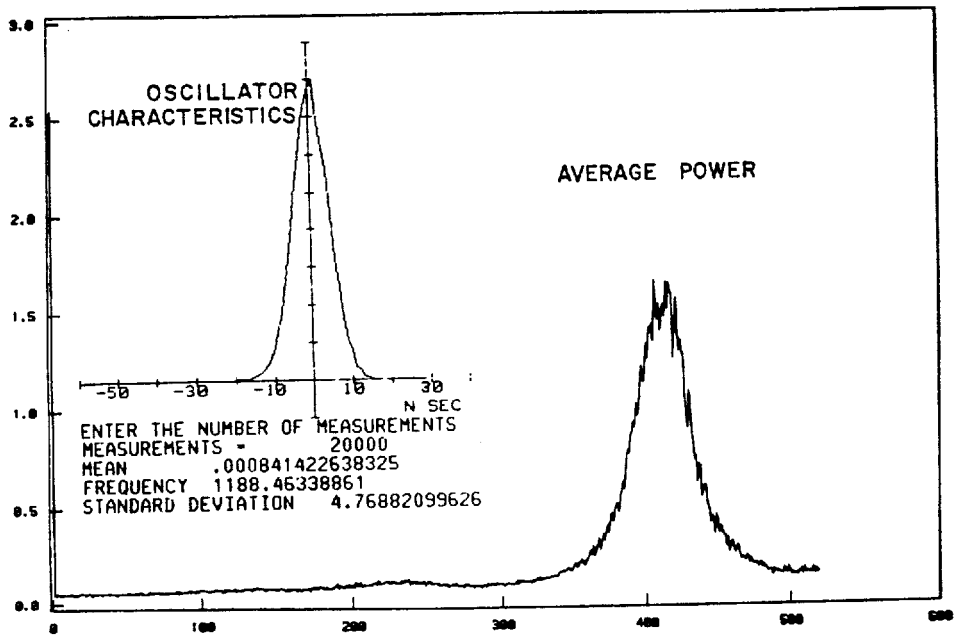


FIGURE 11

samples) in the data toward either a "one" or a "zero". Figure 12 shows the data after adjacent bits are exclusive-ored and scrambled. Note, a d.c. component is still present. This component is removed with additional scrambling of bits. In actual IC operation, each register in the scrambler will contain the exclusive-or sum of a minimum of nine samples before the user can remove a random number from the device. This amount of scrambling removes the d.c. component seen in Figure 12. Figure 13 shows the effect of scrambling five bits without first exclusive-oring adjacent terms. Note that there is no d.c. term in this spectrum. This indicates that the IC would perform better if adjacent bits were not exclusive-ored before they were fed into the scrambler.

#### RNG INTEGRATED CIRCUIT FEATURES

1. 8 bit bidirectional Data Bus.
2. Separate Read ( $RD\sim$ ), Write ( $WR\sim$ ) and Chip Select ( $CS\sim$ ) inputs.  
Note: The  $\sim$  is used here and in the following text to designate the complement or inverse of a signal.
3. 3 bit input Address Bus.
4. Generation of a 536 bit random number accessible in sixty-seven, eight bit bytes.
5. Elementary randomness check via internal 4 bit "run-up" test, during which time a random number is accumulating in the scrambler. External access to statistics generated, i.e., not just a pass fail test. "Run-up" test limits programmed through the host processor. This test may be used to verify the internal circuitry.
6. Internal verification that the data generated and stored in the RNG is the same as the data appearing on the data bus during a microprocessor read of the device.
7. Use of on-chip oscillators or external signals.
8. Output flags, Data Ready and Alarm, may be read from the data bus or on independent output pins. This enables either processor interrupt or processor polled systems to be configured.

## AVERAGE POWER

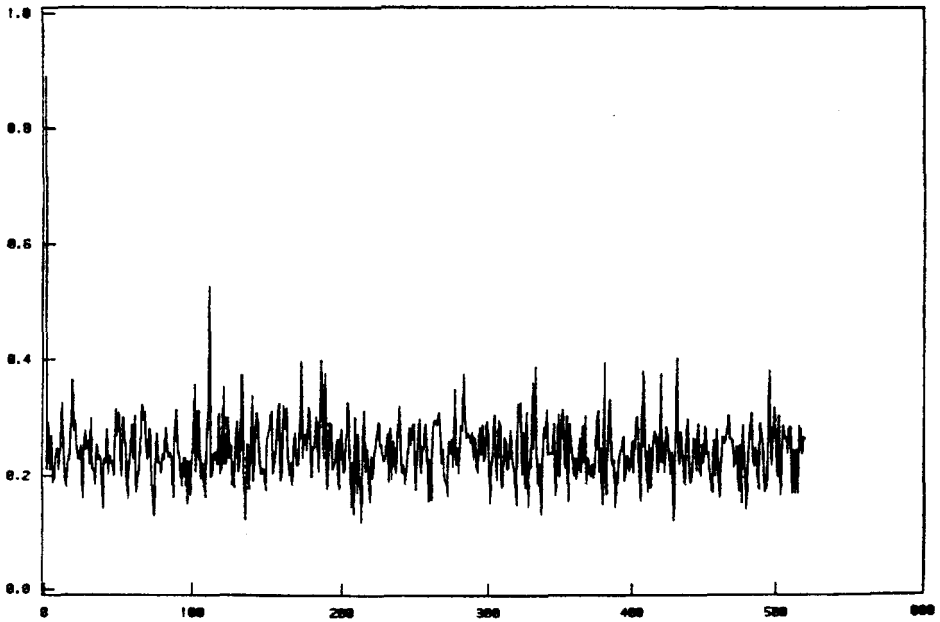


FIGURE 12

## AVERAGE POWER

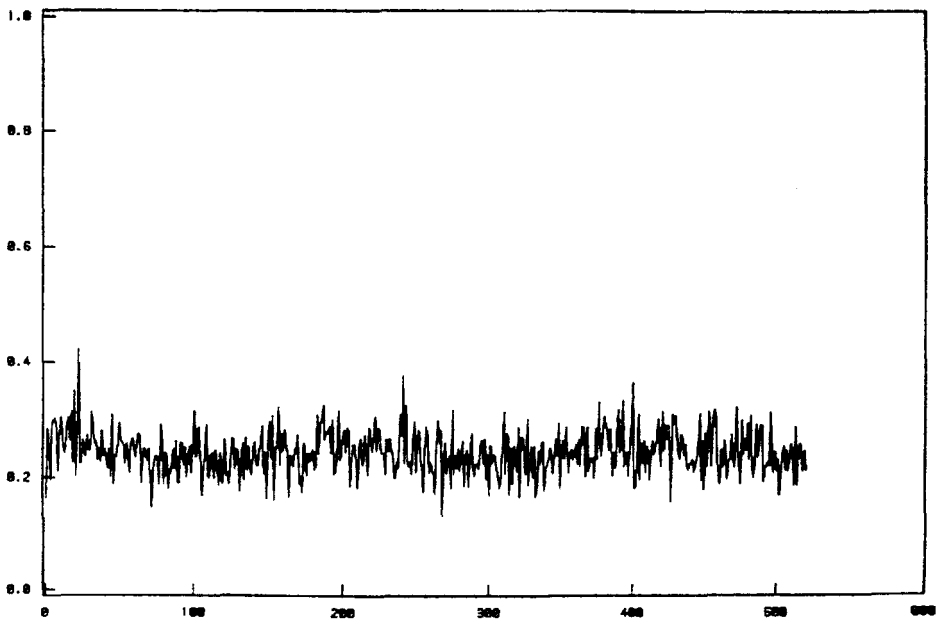


FIGURE 13

## RNG OPERATION

The block diagram of the RNG is shown in Figure 14. The device is configured to appear as a standard microprocessor peripheral. There are eight internal registers for control and/or status reporting. All of these registers may be read. The Upper and Lower Limit Registers, used in a "run-up" test, plus the Status/Command Register control device operation and must be written to, by the host processor, before proper operation can begin.

Following device initialization by the controlling processor, random number generation is initiated with a master reset pulse. Ensuing this master reset, random bits are generated and fill the 536 bit (67 byte) Random Data Byte Register. Following the initial fill, a "run-up" test (described below) is executed on the last four bits in the shift register. If the test passes, the Data Ready flag goes active and the host processor can address the Random Data Byte Register to read the 67 bytes. At the end of the sixty-seventh read pulse, the device automatically behaves as if a new master reset pulse was received. If the "run-up" test fails, the Data Ready flag remains inactive, the Alarm flag goes active and any attempt to read the random data bytes is inhibited. The alarm condition remains active until a Master Reset is issued. Access to the Random Data Byte Register continues to be denied until a "run-up" test is passed.

During any read operation of the random data bytes, the number output to the data bus is checked against the number stored in the shift register and a Bus Error flag goes active if there is any discrepancy. This flag state may be read directly from the Status/Command Register or on the output Alarm pin; it indicates a catastrophic condition due to hardware failure.

## RUN-UP AND SELF TEST

During a run-up test the last four bits in the shift register are compared to the output from a 4 bit Pattern Counter. If both bit patterns match, an Event Counter is incremented. At the end of one thousand non-overlapping four bit tests on a fixed pattern, the result accumulated in the Event Counter is compared to the contents of the Upper and Lower limit registers. If the event count is outside the stored limits the test fails. It takes approximately 20 seconds to



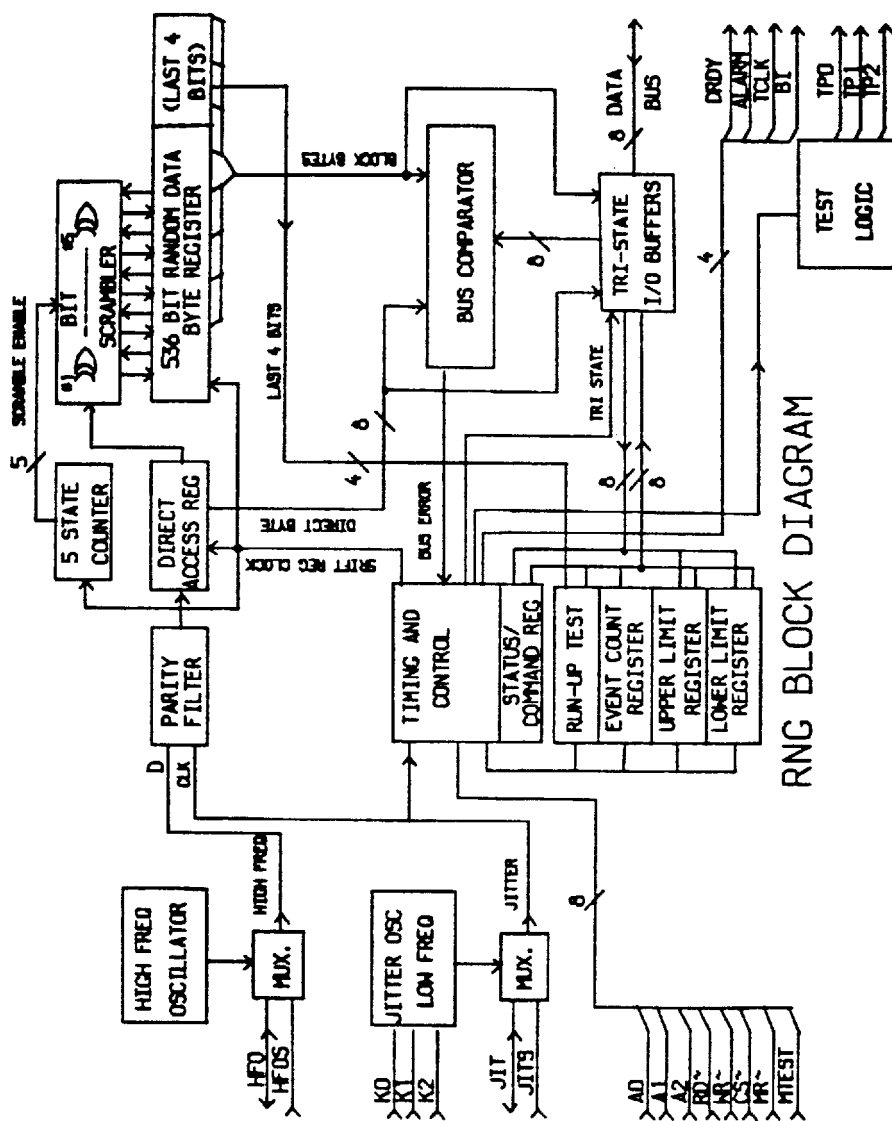


FIGURE 14

execute a "run-up" test with the low frequency jitter oscillator set to 1 KHz. If a different frequency is used, be it from the internal or an external source, the testing time may be computed by multiplying the Jitter oscillator period by 20,096. At the end of a "run-up" test, every scrambler register location will be the exclusive-or sum of either nine or ten data bits coming from the parity filter.

If the random bits are independent with  $P(1)=P(0)=0.5$  the probability of any 4 bit pattern is simply 0.0625 and the expected event count for 1000 samples is 62.5. The event count has a binomial distribution with a standard deviation of 7.65. Using the normal approximation to the binomial distribution, the probability of a test failing can readily be computed. For example, if the limits are set at plus and minus twice the standard deviation, the probability of the test failing is 0.0456 (taken from a table of values of the Standard Normal Distribution Function). At plus and minus three times the standard deviation, the probability of the test failing is 0.0026. In the device structure, the upper and lower limits are entered and appear in registers as hex integers. This test will distort the random numbers coming out of the device by eliminating patterns having very low probability. To prevent this, the upper limit register can be loaded with FF (255) and the lower limit register with 00. With these limits, the test will never fail. The run-up test may be used to completely test the deterministic circuitry by using the "Alarm Test" circuitry explained in the next section.

At this point it is worth talking about oscillator failure and its detection. If the low frequency oscillator were to fail, device operation would halt since that oscillator is used to clock the entire chip. The Data Ready flag would never go active. If the high frequency oscillator were to fail either high or low, the output of the parity filter would be all zeroes. Hence, to check for a high frequency failure the run-up test may be used with the limits set to 01 and FE (254). A master reset must be issued before running the test in order to set the scrambler register to zero. If the oscillator were stuck, any of the 16 possible patterns would fail. The chance of a fully operating device failing under these conditions is

$$1 - 6.22 \times 10^{-16}, (8\sigma)$$

Therefore, it may be a good idea to normally operate the device in this manner.

## REGISTERS

There are eight addressable registers. Figure 15 defines data port output during a register read operation.

## RANDOM NUMBER GENERATOR READ OPERATION

ADDRESS A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	MSB	REGISTER	LSB						
		RANDOM DATA BYTE							
0 0 0		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
		STATUS / COMMAND							
0 0 1		0	0	BUS ERROR	ALARM	ALARM TEST	FREE RUN	MSTR RESET	<u>DRDY</u>
		EVENT COUNT							
0 1 0		T <sub>7</sub>	T <sub>6</sub>	T <sub>5</sub>	T <sub>4</sub>	T <sub>3</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
		PATTERN							
0 1 1		0	0	0	0	S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>
		LOWER LIMIT							
1 0 0		L <sub>7</sub>	L <sub>6</sub>	L <sub>5</sub>	L <sub>4</sub>	L <sub>3</sub>	L <sub>2</sub>	L <sub>1</sub>	L <sub>0</sub>
		UPPER LIMIT							
1 0 1		U <sub>7</sub>	U <sub>6</sub>	U <sub>5</sub>	U <sub>4</sub>	U <sub>3</sub>	U <sub>2</sub>	U <sub>1</sub>	U <sub>0</sub>
		RANDOM DATA BYTE COUNTER							
1 1 0		CD <sub>7</sub>	CD <sub>6</sub>	CD <sub>5</sub>	CD <sub>4</sub>	CD <sub>3</sub>	CD <sub>2</sub>	CD <sub>1</sub>	CD <sub>0</sub>
		DIRECT ACCESS							
1 1 1		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

Figure 15

Figure 16 defines data port input during a write operation.

## RANDOM NUMBER GENERATOR WRITE OPERATION

ADDRESS	REGISTER							
$A_2 \ A_1 \ A_0$	MSB				LSB			
0 0 0								
	STATUS / COMMAND							
0 0 1					ALARM TEST	FREE RUN	MSTR RESET	
0 1 0								
0 1 1								
	LOWER LIMIT							
1 0 0	$L_7$	$L_6$	$L_5$	$L_4$	$L_3$	$L_2$	$L_1$	$L_0$
	UPPER LIMIT							
1 0 1	$U_7$	$U_6$	$U_5$	$U_4$	$U_3$	$U_2$	$U_1$	$U_0$
1 1 0								
1 1 1								

Figure 16

1. Address 0 selects the read only Random Data Byte Register. Sixty-seven read pulses empties the shift register. After emptying the shift register the "run-up" test cycle begins again. If less than 67 read pulses are input, the device will remain inactive waiting for the remaining pulses unless a Master Reset is issued. Addressable register 6 is a down counter and keeps track of the number of random bytes remaining in the shift register.
2. Address 1 selects the read/write Status/Command Register and must be written to, before certain device operations can begin.
  - A. Bit 0 of the Status/Command Register is a read only, active low, Data Ready flag (DRDY $\sim$ ) used to indicate a "run-up" test has passed and a 67 byte random number is stored in the Random Data Byte Register (Address 0). This flag remains active until: all 67 random data bytes have been read, a bus error is detected, or a Master Reset is issued. This information is also available on the Data Ready output pin.
  - B. Bit 1 of the Status/Command Register is a read/write, active high, Master Reset command. If active, a Master Reset condition exists until this bit is cleared by pulsing the external Master Reset pin or by writing a "0" to this bit. A master reset clears the scrambler.
  - C. Bit 2 of the Status/Command Register is a read/write, active high, Free Run command (FR). In the inactive state, the device executes a single "run-up" test and halts operation. If the test passes, the Data Ready flag goes active and all 67 random data bytes must be read (or a Master Reset issued) before a second "run-up" test is begun. If Bit 2 is set active, the device continually executes "run-up" tests. The Data Ready flag will go active after the first "run-up" test passes and remain active until a failure. Once this flag has gone active, data can be read from the Random Byte Register. During the reading of the Random Byte Register, "run-up" tests temporarily cease. That is, with the first read pulse accessing the Random Byte Register, the current "run-up" test halts. After the sixty-seventh read pulse which

empties the Random Byte Register, "run-up" tests start anew. When a failure occurs the Alarm flag goes active, the Data Ready flag goes inactive, and any attempt to read from the Random Byte Register is inhibited. An Alarm condition can only be cleared by a Master Reset. Bit 2 can be set at any time during chip operation. An external Master Reset pulse clears bit 2, but an internal Master Reset has no affect.

- D. Bit 3 of the Status/Command Register is a read/write, active high, Alarm Test command (ALRMT). If active, a known sequence of zeros and ones is automatically loaded into the Random Byte Register producing known pattern counts for the "run-up" test. The counts which are generated are given in Table 3.

Alarm Test Register Data							
Pattern (Hex)	Event Count (Hex)	Random Data Byte Register					
		Byte Number	Byte (Hex)	Byte Number	Byte (Hex)	Byte Number	Byte (Hex)
0	A0	1	71	23	E8	45	C9
1	21	2	F0	24	A1	46	61
2	4E	3	EC	25	C8	47	B0
3	25	4	A1	26	73	48	E0
		5	58	27	B0	49	A1
4	55	6	73	28	E8	50	C9
5	3B	7	F0	29	A1	51	61
6	15	8	EC	30	C8	52	B0
7	4E	9	A1	31	73	53	A2
8	28	10	58	32	B0	54	A1
9	52	11	73	33	E0	55	C9
A	42	12	F0	34	A1	56	61
B	38	13	E8	35	C8	57	B0
C	1D	14	A1	36	73	58	A2
D	3F	15	58	37	B0	59	A1
E	47	16	73	38	E0	60	C9
F	2A	17	B0	39	A1	61	41
		18	E8	40	C8	62	B0
		19	A1	41	61	63	A2
		20	58	42	B0	64	A9
		21	73	43	E0	65	C9
		22	B0	44	A1	66	41
						67	B0

Table 3

This test is used to check the "run-up" and alarm circuitry. It can also be used to produce a known pattern

in the Random Byte Register which can be read out if desired. This pattern is also listed in Table 3. In order for this test to operate correctly the device must be first cleared with the internal Master Reset command (set bit 1). Then, on a subsequent write cycle, the internal master reset command bit must be cleared simultaneously with the Alarm Test bit being set. Bit 3 is cleared with an external Master Reset (internal Master Reset has no affect).

- E. Bit 4 of the Status/Command Register is a read only, active high, Alarm flag (ALRM). If active, a "run-up" test has failed. An active Alarm flag will inhibit device operation and can only be cleared by a Master Reset. This information is also indicated by an active Alarm flag pin.
- F. Bit 5 of the Status/Command Register is a read only, active high, Bus Error flag (BE). If active, there has been a discrepancy between the data in the Random Data Byte Register and the data appearing on the eight bit bidirectional Data Bus during a read of the register. An active Bus Error flag will inhibit device operation and can only be cleared by a Master Reset. This information is also indicated by an active Alarm flag pin.
- G. Bits 6 and 7 of the Status/Command Register are unused and always low.

- 3. Address 2 selects the read only Event Count Register. This register stores the hex event count from the most recently completed "run-up" test. This is an eight bit register and the maximum count it can display is decimal 255. A reading less than 255 (hex FF) indicates the actual event count obtained during the last "run-up" test while a reading of 255 indicates an event count of 255 or more.
- 4. Address 3 selects the read only Pattern Register. This register stores the 4 bit hex pattern associated with the most recently completed "run-up" test. At the completion of every successful "run-up" test, the pattern counter is incremented. If the "run-up" test fails, the pattern is not changed and the test is repeated following a Master Reset. Master Reset does not affect this counter. In power up this counter assumes an arbitrary

state and increments from there.

5. Address 4 selects the read/write Lower Limit Register. This register stores the hex lower limit associated with the "run-up" test and must be written to before proper operation of the device can begin.
6. Address 5 selects the read/write Upper Limit Register. This register stores the hex upper limit associated with the "run-up" test and must be written to before proper operation of the device can begin.
7. Address 6 selects the read only Random Data Byte Counter Register. This down counter register keeps track of the number of random bytes left in the shift register (Address 0). Following an active Data Ready signal, this register is preset to hex 43 (67 decimal). After the 67 random data bytes have been read, this register is at hex 0 and remains there until the next active Data Ready signal.
8. Address 7 selects the read only Direct Access Register. This register along with the TCLK pin allows the user to continually monitor the random data byte at the input to the scrambler. The random data is latched into this register on the rising edge of the TCLK signal and it can be read after this edge. If the internal random bit generator is used at 1KHz, the TCLK period is approximately 32 milliseconds. Using a different jitter frequency or an external source, the TCLK period may be computed by multiplying the Jitter oscillator period by 32.

Data read from this register has not been subjected to the bit scrambler and may have high bit to bit correlation. Thus, this data should not be used in place of data obtained from the Random Byte register as a random number but should only be used for device monitoring and/or testing.

#### PIN DESCRIPTION

Figure 17 shows the RNG pin configuration.

1. VDD This is the +5 volt power supply input.
2. GND Ground.



3.  $\overline{MR}$  Master Reset is used to clear the RNG when the Chip Select line is active. The control bits in the Status/Command Register (Address 1) are all set inactive, the Event Count Register is set to zero, the scrambler is cleared, and a test sequence is begun on the rising edge of the reset pulse. The Pattern Register (Address 3), Lower Limit Register (Address 4), and Upper Limit Register (Address 5) are unaffected.

## RANDOM NUMBER GENERATOR PIN DIAGRAM

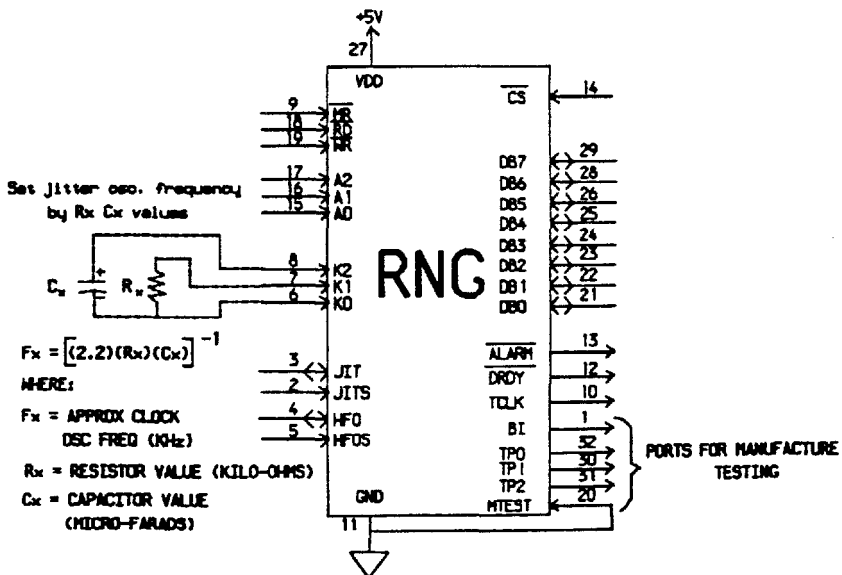


Figure 17

4.  $\overline{RD}$  Read is an active low input used with the Chip Select, the Address Bus, and the Data Bus to read one of the eight internal registers. The data appears on the bus following the falling edge of the pulse and remains on the bus as long as  $\overline{RD}$  is low. The Write input should be held inactive during a Read pulse.
5.  $\overline{WR}$  Write is an active low input used with the Chip Select, the Address Bus, and the Data Bus to write to one of three internal registers. The data is latched into the addressed register on

the rising edge of the Write pulse. The Read input should be held inactive during a Write pulse.

6. A2 - A0 The Address Bus input is used to select an internal register for a read or write operation.
7. DB7 - DB0 The Data Bus, an eight bit bidirectional port, is used to read data from or write data to the internal registers. The output buffers driving the eight bit bus are in a high impedance state if either CS $\sim$  or RD $\sim$  are inactive.
8. CS $\sim$  Chip Select is an active low input. When active MR $\sim$ , RD $\sim$ , and WR $\sim$  are enabled; when inactive these inputs are disabled.
9. ALARM $\sim$  Alarm flag is an active low output used to indicate either a "run-up" test failure or a bus error. Both of these conditions may also be read from the Status/Command Register (Address 1). An active Alarm flag may only be cleared by a Master Reset.
10. DRDY $\sim$  Data Ready flag is an active low output used to indicate a "run-up" test has passed and a 67 byte random number is stored in the Random Data Byte Register. This flag may also be read from the Status/Command Register (Address 1). Following the rising edge of the sixty-seventh RD $\sim$  pulse, with the Random Data Register addressed (Address 0), the DRDY $\sim$  flag goes inactive.
11. TCLK Test Clock is an output used with the Direct Access Register to monitor the random data at the input to the Random Data Byte Register. The random data is latched into the Direct Access on the rising edge of the TCLK signal and it can be read after this edge. Using the internal random bit generator (with jitter osc. set to 1KHz), the TCLK period is approximately 32 milliseconds. Using a different oscillator frequency or an external source, the TCLK period may be computed by multiplying the Jitter oscillator (JIT) Register period by 32.
12. HFOS This pin controls a switch that determines the high frequency source for the chip: when HFOS is "0", the internal high frequency oscillator is used (about 8MHz); when HFOS is 1, an external oscillator is expected at the HFO pin. It is recommended that an external high frequency square wave be used for critical applications (possibly the system clock).

13. HFO This pin is bi-directional and serves two purposes: when HFOS is "0", HFO serves as an output for viewing the internal high frequency oscillator; when HFOS is 1, HFO serves as an input for an external high frequency oscillator.
14. JITS This pin controls a switch that determines the jitter input source for the chip: when JITS is "0", the internal jitter oscillator is used; when JITS is 1, an external oscillator is expected at the JIT pin.
15. JIT This pin is bi-directional and serves two purposes: when JITS is "0", JIT serves as an output for viewing the internal jitter oscillator when JITS is 1, JIT serves as an input for an external jitter oscillator.
16. BI This pin monitors the output of the first sampling D-type flip-flop at the front end of the parity filter. HFO is the data into the positive-edge-triggered flip-flop and JIT is the clock.
17. K0, K1, K2 These leads are used to attach the external resistor and capacitor which control the frequency of oscillation of the on-chip jitter oscillator. The frequency of oscillation is approximately given by

$$f = 1/(2.2RC)$$

The resistor is connected between K0 and K1 while the capacitor is connected between K0 and K2.

18. TEST PINS used for manufacture purposes: MTEST This pin should always be grounded when in normal use. This is only used for manufacturing tests. TP0, TP1, TP2 These are only used for manufacturing tests and should remain floating since they are outputs.

## CONCLUSIONS

An LSI CMOS random number generator which generates a truly random binary number has been described. The fundamental mechanism generating the random bit stream is based on a previously documented physical phenomenon and this paper has tried to quantify the magnitude of the parameters governing device performance. Statistical tests

have been run on device output and no problems have been observed for sampling oscillator frequencies ( $F_c$ ) below 2 KHz. Although it takes almost 20 seconds to generate a 67 byte number when a 1000 Hz sampling clock is used, the generation can be done in the background after power-up or during a session so that keys and/or initial values are available on request. This is the first integrated device of its type that we are aware of and it should solve the problem of generating cryptographic keys and/or initial values in cryptographic systems.

#### REFERENCES

- [1] The RAND Corporation, "A Million Random Digits with 100,000 Normal Deviates", The Free Press, New York.
- [2] Motorola Data Book, "Phase-Locked Loop Systems", Motorola Semiconductor Products Inc., second edition, August, 1973.
- [3] Asad A. Abidi and Robert G. Meyer, "Noise in Relaxation Oscillators", IEEE Journal of Solid-State Circuits, Vol. SC-18, No 6, December 1983.

# GENERALIZED LINEAR THRESHOLD SCHEME

S.C. Kothari  
Department of Computer Science  
Iowa State University  
Ames, Iowa 50011

## ABSTRACT

A generalized linear threshold scheme is introduced. The new scheme generalizes the existing linear threshold schemes. The basic principles involved in the construction of linear threshold schemes are laid out and the relationships between the existing schemes are completely established. The generalized linear scheme is used to provide a hierarchical threshold scheme which allows multiple thresholds necessary in a hierarchical environment.

## INTRODUCTION:

The protection of important information is an age old problem. In any scheme devised to protect information, care has to be taken to ensure that the information does not get lost, destroyed, or into wrong hands, and at the same time the scheme should be efficient. A simple solution to protect the information from loss or destruction, is to make multiple copies of the information and distribute the copies. But with multiple copies the probability that the information will get into wrong hands, increases and the simple solution becomes unacceptable. The question of protection of information has received a lot of attention in recent years because of the proliferation of computers into areas such as electronic mail, electronic fund transfer, and storage of information.

There have been several schemes, called cryptosystems, developed in the past to protect information. A very important and interesting class of cryptosystems called the public key cryptosystems came into existence in the seventies. The important concept underlying the public key cryptosystems is to create a cryptosystem such that the knowledge of the encoding key does not lead to the computation of the decoding key in a reasonable amount of computer time. This enables the public key cryptosystems to make the encoding key public and also solve the problem of electronic signature. The reader is referred to [Diffie 76], [Rivest 78] for discussion of the public key cryptosystems. In 1979, a different type of protection scheme, called the threshold scheme<sup>1</sup> was introduced independently by Blakely and Shamir. The important idea underlying the threshold scheme is to create "shadows"<sup>2</sup> of the message (secret) such that

---

<sup>1</sup>The other commonly used names for threshold schemes are key safeguarding schemes and secret sharing schemes.

<sup>2</sup>The term "shadows" was originally introduced by Professor Blakely in [Blakely 79].

unless a certain number (called the threshold) of "shadows" are not available the message (secret) cannot be retrieved. Discussion of the threshold schemes is found in [Blakely 79], [Shamir 79]. Several other threshold schemes have been introduced since then.

This paper focuses on four of the existing threshold schemes [Blakely 79], [Shamir 79], [Bloom], [Karnin 83]. It is shown that the four schemes are founded on common principles derived from linear algebra and for this reason we will refer to these four schemes as linear threshold schemes. A generalized linear threshold scheme which subsumes the various threshold schemes is presented. The generalized threshold scheme extracts the essence of the linear threshold schemes, making transparent the basic principles.

Roughly speaking a generalized linear threshold scheme works as follows. A secret is represented by a scalar and a linear variety is chosen to conceal the secret. A linear functional fixed in the beginning, and known to all trustees is used to reveal the secret from the linear variety. The  $n$  shadows are hyperplanes containing the linear variety. Moreover the hyperplanes are chosen to satisfy the condition that the intersection of less than  $t$  ( $t \leq n$ ) of them results in a linear variety which projects uniformly over the scalar field by the linear functional used for revealing the secret. The number  $t$  is called the threshold. Thus as more shadows are known more information is revealed about the linear variety used to keep the secret, however, no information is revealed until the threshold number of shadows are known.

Karnin et al show in [Karnin 83] that Shamir's, and Blakeley's schemes are special cases of their threshold scheme. It is shown here that with the exception of Shamir's scheme the remaining three threshold schemes are equivalent to each other and explicit algorithms are presented to convert one scheme to another. The Shamir's scheme is a specialization of the remaining three schemes and all the four schemes are specializations of the generalized linear threshold scheme. Also a much simpler proof of perfect security compared to [Blakely 81] is presented. This proof nicely explains the common mechanism used for perfect security in various linear threshold schemes.

The generalized linear threshold scheme allows linear varieties of positive dimension to conceal the secret. This fact is utilized in constructing a hierarchical threshold scheme. The hierarchical threshold scheme uses a chain of linear varieties to keep a secret and allows multiple thresholds for hierarchy of trustees.

## 2. DEFINITIONS AND PRELIMINARY RESULTS

In this section some preliminary results and definitions are discussed. This material will be used in the construction of the generalized threshold scheme and also in the proofs to show that other threshold schemes are specializations of the generalized threshold scheme. Some of these are standard results but they are included here for the sake of completeness and to fix the notation. The interested readers may look at [Kuiper 65] for further discussion.

DEFINITION: A threshold scheme is a process which converts a given number  $x$  called the "message" to  $n$  other numbers  $y_i$ 's called the "shadows" which satisfy the property: there exists a number  $t$  ( $t \leq n$ ) called the threshold such that  $x$  can be retrieved if any  $t$  of the  $n$  "shadows" are known, but less than  $t$  "shadows" reveal no information about the message. More specifically such a threshold scheme is called  $t$  out of  $n$  threshold scheme.

Using the entropy function  $H$  from [Shannon 48] we can state the requirements in the threshold scheme as

$$(i) H(x \mid y_{i_1}, y_{i_2}, \dots, y_{i_t}) = 0$$

$$(ii) H(x) = H(x \mid y_{i_1}, y_{i_2}, \dots, y_{i_{t-1}})$$

for an arbitrary set of  $t$  indices  $\{i_1, i_2, \dots, i_t\}$ .

Let  $k$  denote a finite field and  $k^n$  denote the set consisting of  $n$ -tuples over  $k$ . The set  $k^n$  is a vector space over  $k$  of dimension  $n$  in a natural way. The set  $k^n$  is also called an affine space over  $k$  and the individual  $n$ -tuples are referred to as the points of the affine space.

DEFINITION: A subset  $S$  of  $k^n$  is called an affine variety of  $k^n$  if there exist a finite set  $f_i(x_1, x_2, \dots, x_n)$ ,  $i = 1, 2, \dots, m$ , of polynomials in  $n$  variables such that

$$S = \{(a_1, a_2, \dots, a_n) \in k^n \mid f_i(a_1, a_2, \dots, a_n) = 0 \text{ for } i=1, 2, \dots, m.\}$$

The equations  $f_i(x_1, x_2, \dots, x_n) = 0$  are called the defining equations of the affine variety  $S$ .

DEFINITION: An affine variety is called a linear variety if all its defining equations are linear.

DEFINITION: A linear variety is called a homogenous linear variety if all its defining equations are homogeneous.

Given a linear polynomial  $f(x_1, x_2, \dots, x_n) = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$ , we represent it by the vector  $(b_0, b_1, \dots, b_n)$  in  $k^{n+1}$ , representing the coefficients of  $f$ . We will identify a linear polynomial  $f(x_1, x_2, \dots, x_n)$  with the vector representing its coefficients.

DEFINITION: Given a linear variety  $S$ , define

$$E(S) = \{f \in k^{n+1} \mid f(a_1, a_2, \dots, a_n) = 0 \text{ for all } (a_1, a_2, \dots, a_n) \in S\}.$$

$E(S)$  is a vector subspace of  $k^{n+1}$ .

DEFINITION: Given subsets  $S$  and  $W$  of  $k^n$  and vector  $c$  in  $k^n$  define  $S = c+W$  if

$$S = \{v \in k^n \mid v = c+w \text{ for } w \in W\}.$$

The function  $\dim(\ )$  is used to denote the dimension of a vector space.

LEMMA 1: Given a linear variety  $S$  of  $k^n$ , there exists a vector  $c = (c_1, c_2, \dots, c_n)$  and a vector subspace  $W$  of  $k^n$  such that

$$(i) \quad S = c + W$$

and

$$(ii) \quad \dim(W) + \dim(E(S)) = n.$$

Proof: The proof follows from the Gaussian elimination process and other standard arguments from vector space theory.

DEFINITION: For a linear variety  $S = c+W$ , define  $\dim(S)$  to be  $\dim(W)$ .

NOTATION: Given a linear variety  $S$ , the notation  $S = w + W$  indicates that  $w$  is a vector and  $W$  is a vector subspace of  $k^n$ .

LEMMA 2: Let  $S_1 = w_1 + W_1$ , and  $S_2 = w_2 + W_2$  be linear varieties of  $k^n$ . Then  $S_1 \cap S_2$ , is either empty or else it is a linear variety such that  $S_1 \cap S_2 = w + W$  where  $W$  contains  $W_1 \cap W_2$ .

Proof: The proof follows from standard vector space arguments.

DEFINITION: Given vector subspaces  $W_1$  and  $W_2$  of  $k^n$ ,  $W_1 + W_2$  is defined as the vector space where

$$W_1 + W_2 = \{v \in k^n \mid v = w_1 + w_2 \text{ for } w_1 \in W_1 \text{ for } i=1,2.\}$$

The following is a standard result from the vector space theory.

LEMMA 3: If  $W_1$  and  $W_2$  are vector spaces of  $k^n$  then

$$\dim(W_1 + W_2) = \dim(W_1) + \dim(W_2) - \dim(W_1 \cap W_2).$$

DEFINITION: If  $S$  is a linear variety of  $k^n$  such that  $\dim(E(S)) = 1$  then  $S$  is called a hyper plane.

LEMMA 4: Let  $S = w + W$  be a linear variety and  $t = \dim(E(S))$ . Let  $H_1, H_2, \dots, H_m$  be hyperplanes containing  $S$ . Then,

$$(i) \quad \text{If } m < t \text{ then } \bigcap_{i=1}^m H_i \text{ strictly contains } S,$$

$$(ii) \quad \text{If } \bigcap_{i=1}^m H_i = S \text{ then } m \geq t.$$

Proof: Note that (ii) clearly follows from (i) because  $H_i$  contains  $S$  for  $i=1, 2, \dots, m$ . Let  $T = \bigcap_{i=1}^m H_i$ . Clearly  $E(T) = E(H_1) + E(H_2) + \dots + E(H_m)$ . By repeated applications of



Lemma 3 it follows that  $\dim(E(T)) \leq m$ . If  $m < t$  then it follows from Lemma 1 that  $\dim(S) < \dim(T)$ , thus  $T$  strictly contains  $S$ .

DEFINITION: Let  $S$  be a linear variety and  $t = \dim(E(S))$ . The hyperplanes  $H_1, H_2, \dots, H_m$  for  $m \geq t$ , are said to be in general position with respect to  $S$ , if the intersection of any  $t$  of them is  $S$ .

DUALIZATION PRINCIPLE: Let  $V$  be a vector space of dimension  $n$  over the base field  $k$ . Let  $\tilde{V}$  be the set of linear functionals on  $V$  then  $\tilde{V}$  itself forms a vector space of dimension  $n$  called the dual space of  $V$ . The space  $\tilde{V}$  can be identified with  $k^n$  as follows:

Fix a basis of  $V$ . Then for every linear functional  $L$  on  $V$  there exist a unique vector  $(a_1, a_2, \dots, a_n)$  in  $k^n$  such that for every  $v$  in  $V$ ,

$$L(v) = a_1 v_1 + a_2 v_2 + \dots + a_n v_n,$$

where  $(v_1, v_2, \dots, v_n)$  is the representation of  $v$  with respect to the fixed basis of  $V$ .

For every  $L$  belonging to the dual space  $\tilde{V}$  we identify it with the vector  $(a_1, a_2, \dots, a_n)$  in  $k^n$  as described above.

DEFINITION: Given a vector  $v$  belonging to a vector space  $V$  and an element  $a$  of  $k$ , define

$$H(v, a) = \{L \in \tilde{V} \mid L(v) = a\}.$$

LEMMA 5: Given a vector  $v$  in  $V$  and an element  $a$  in  $k$  the set  $H(v, a)$  is a hyperplane in  $\tilde{V}$ .

Proof: Let  $v = (v_1, v_2, \dots, v_n)$  be the representation of  $v$  with respect to a fixed basis of  $V$ . Then by the dualization principle  $H(v, a)$  can be identified with the set

$$\{(a_1, a_2, \dots, a_n) \in k^n \mid a_1 v_1 + a_2 v_2 + \dots + a_n v_n = a\}$$

Thus  $E(H(v, a))$  is a vector space of dimension one, generated by the vector  $(-a, v_1, v_2, \dots, v_n)$  and so  $H(v, a)$  is a hyperplane.

LEMMA 6: Let  $V$  be a vector space of dimension  $n$ . Let  $v_i$  for  $i=1, 2, \dots, m$ ,  $m \geq n$ , be vectors in  $V$  such that any  $n$  of them are linearly independent. Let  $L$  be a linear functional on  $V$  and let  $L(v_i) = a_i$  for  $i=1, 2, \dots, m$ . Then the hyperplanes  $H_i = H(v_i, a_i)$  are in general position with respect to  $(L)$ .

Proof: Since  $L(v_i) = a_i$ ,  $L$  belongs to  $H_i$  for  $i=1, 2, \dots, m$ . By lemma 1,  $\dim(E(\{L\})) = n$ . For  $i=1, 2, \dots, m$ , by  $(v_i, a_i)$  we denote the vector in  $k^{n+1}$  whose projection on the first

$n$  components is given by the vector  $v_i$  and the  $(n+1)$ -th component is  $a_i$ . Any of the  $n$  vectors  $(v_i, a_i)$  for  $i=1,2,\dots,m$  are linearly independent because the same property is true for their projections  $v_i$ 's by assumption. To prove that the hyperplane  $H_i$  are in general position we need to show that intersection of any  $n$  of them is  $\{L\}$ .

Let  $S = \bigcap_{i=1}^n H_i$ . Without loss of generality it is enough to show that  $S = \{L\}$ .

By lemma 2,  $S$  is a linear variety.  $L$  belongs to  $S$  because  $L$  belongs to every hyperplane  $H_i$  for  $i=1,2,\dots,m$ . The vectors  $(v_i, a_i)$  belong to  $E(S)$  for  $i=1,2,\dots,n$ . Since the vectors are linearly independent it follows that  $\dim(E(S))=n$ . Then by lemma 1,  $\dim(S) = 0$  and so  $S = \{L\}$  since  $L$  belongs to  $S$ .

LEMMA 7: Let  $v_i = (1, b_i, b_i^2, \dots, b_i^{n-1})$  be vectors belonging to  $k^n$  for  $i=1,\dots,m$  where  $m \geq n$  and  $b_i \neq b_j$  for  $i \neq j$  then any  $n$  of the  $m$  vectors  $v_i$ 's are linearly independent.

Proof: Without loss of generality it is enough to show that  $v_i$ 's for  $i=1,2,\dots,n$  are linearly independent. Let  $B$  be the  $n \times n$  matrix such that its  $(i,j)$ -th entry is  $b_i^{j-1}$ . The matrix  $B$  is known as a Vandermonde matrix. It is a property of a Vandermonde matrix that  $\det(B) \neq 0$  if and only if  $b_i \neq b_j$  for  $i \neq j$ . Thus if  $b_i \neq b_j$  for  $i \neq j$  then  $\det(B) \neq 0$  which implies that  $v_i$ 's are linearly independent for  $i=1,2,\dots,n$ .

Definition: Given a linear functional  $f$  and an element  $a$  belonging to  $k$  define  $H(f,a) = \{v \in V \mid f(v) = a\}$ .

LEMMA 8:  $H(f,a)$  is a hyperplane in  $V$ .

Proof: This is a dual of lemma 5.

The following lemma is the mathematical basis for the perfect security of the various linear threshold schemes.

LEMMA 9: Let  $f$  be a linear functional on a vectorspace  $V$ .  $T$  is a linear subvariety of  $V$  such that  $T$  is not contained in  $H(f,a)$  for any  $a$  belonging to  $k$ . Then  $f$  uniformly projects  $T$  over  $k$ .

Proof: Since  $T$  is not contained in any  $H(f,a)$ ,  $f$  projects  $T$  onto  $k$ .

Let  $T_a = \{v \mid v \in T \text{ and } f(v) = a\}$ . Then  $T_a = v_a + W$  where  $v_a$  is a vector in  $T$  which projects to  $a$  and  $W$  is subspace contained in the kernel of  $f$  and it does not depend on  $a$ . Thus cardinality of  $T_a$  which is the same as the cardinality of  $W$  is independent of  $a$  i.e.  $f$  projects  $T$  uniformly over  $k$ .

### 3. DESCRIPTION OF LINEAR THRESHOLD SCHEMES

In this section we briefly describe the threshold schemes due to Blakely, Bloom, Karnin-Greene-Hellman and Shamir. For more detailed descriptions of these schemes refer to [Blakely 79], [Shamir 79], [Bloom], [Karnin 83]. For uniformity of descriptions all the three schemes are set up to give  $n$  "shadows" and the threshold is  $t$  where  $n$  and  $t$  are integers such that  $n \geq t$ .

#### Blakely's Threshold Scheme (affine version):

Blakely's threshold scheme starts with a  $t$  dimensional affine space  $V$ . The key is concealed by specific coordinate of a point  $S$  of  $V$ . The  $n$  "shadows" are given by hyperplanes  $H_i$ ,  $i=1,2,\dots,n$ , of  $V$  such that the  $H_i$  and the specific coordinate plane passing through  $S$  are in general position with respect to  $S$ .

Now given any  $t$  distant "shadows"  $H_i$ , we can get the point  $S$  representing the key by intersecting the "shadows". However if only  $r$  "shadows",  $r < t$ , are known, then by intersecting the  $r$  hyperplanes corresponding to the "shadows" we get  $(t-r)$  dimensional linear variety strictly containing  $S$ . Thus  $S$  cannot be determined and no information about the key is revealed.

#### Bloom's Threshold Scheme:

Bloom's scheme starts with a  $t$  dimensional vector space  $V$ . The  $n$  vectors  $v_i$  for  $i=0,1,\dots,n$ , are chosen such that any  $t$  of them are linearly independent and consequently span the vector space  $C$ . A linear functional  $L$  on  $V$  is chosen such that  $L(v_0)$  represents the key. The "shadows"  $S_i$  for  $i=1,2,\dots,n$  are defined to be  $S_i=L(v_i)$ . Now, given any  $t$  the linear functional  $L$  is completely determined and the key can be computed using  $v_0$ . However if any  $r$  "shadows",  $r < t$ , are known then  $L$  is not completely determined and no information about the key is revealed.

#### Shamir's Threshold Scheme:

Shamir's scheme starts with a polynomial

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1},$$

and nonzero distinct scalars  $b_i$  for  $i=1,2,\dots,n$ . The key  $S$  is represented by  $a_0$ . The  $n$  "shadows"  $S_i$ , for  $i=1,2,\dots,n$  are defined as

$$S_i = f(b_i).$$

Given any  $t$  distinct "shadows"  $f(x)$  can be determined by Lagrange interpolation formula and the key is obtained by evaluating  $f(x)$  at  $x=0$ . If only  $r$  "shadows",  $r < t$ , are known then  $f(x)$  cannot be determined and no information about the key is revealed.

#### Karnin-Greene-Hellman Threshold Scheme:

The scheme starts with  $n+1$  column vectors  $A_0, A_1, \dots, A_n$  of size  $t$  such that any  $t$

of them have full rank.  $U$  is a row vector of size  $t$ . The key is given by  $U \cdot A_0$ . The  $n$  shadows are given by  $U \cdot A_i$  for  $i=1,2,\dots,n$ . If any  $t$  of the  $n$  shadows are known then  $U$  can be determined and the key is obtained by evaluating  $U \cdot A_0$ . If less than  $t$  shadows are known then  $U$  is not determined and no information about the key is revealed.

#### 4. GENERALIZED LINEAR THRESHOLD SCHEME:

A generalized  $t$  out of  $n$  threshold scheme is constructed as follows. Let  $V$  be a  $(d+t)$  dimensional vectorspace. The secret is a scalar  $\alpha$  concealed by a  $d$  dimensional linear subvariety  $S$ . A linear functional  $f$  is used to reveal the secret from  $S$  i.e.  $f$  is chosen such that  $f(v)$  is equal to  $\alpha$  for any  $v$  belonging to  $S$ . The linear variety  $S$  is kept secret but the linear functional  $f$  is made known to all the trustees involved.

The  $n$  shadows are given by  $n$  hyperplanes. The hyperplanes representing the shadows and the hyperplane  $H(f,\alpha)$  together are chosen to be in general position with respect to  $S$ .

Given any  $t$  shadows,  $S$  is obtained by intersecting the corresponding hyperplanes. If less than ( $< t$ ) shadows are known the corresponding hyperplanes intersect in a linear subvariety  $S'$  containing  $S$ . Moreover  $S'$  is not contained in  $H(f,\alpha)$  because the hyperplanes intersecting in  $S'$  and  $H(f,\alpha)$  are in general position by choice. In view of lemma 9  $S'$  reveals no information about  $\alpha$ .

#### 5. INTERRELATION OF LINEAR THRESHOLD SCHEMES:

This section presents conversion algorithms proving that the  $t$  out of  $n$  threshold schemes of Blakely, Bloom and Karnin-Greene-Hellman are equivalent. The same notation that is used to describe the schemes is used again to describe the algorithms.

##### Algorithm BLBM:

The following algorithm converts a Blakely scheme to a Bloom scheme.

1. Let  $e_1, e_2, \dots, e_t$  be the standard basis of  $V$ .  
Choose the linear functional  $L$  such that  $L(e_i)$  is the  $i$ -th coordinate of  $S$  for  $1 \leq i \leq t$ .
2. If the  $k$ -th coordinate of  $S$  is the secret to be concealed then choose  $v_0$  to be  $e_k$ .
3. Choose  $v_i$  to be the vector representing the coefficients of  $H_i$  for  $i=1,2,\dots,n$ .

##### Algorithm BMKGH:

The following algorithm converts a Bloom scheme to a Karnin-Greene-Hellman scheme.

1. Let  $e_1, e_2, \dots, e_t$  be the standard basis of  $V$ . Set  $U = (L(e_1), L(e_2), \dots, L(e_t))$ .

2. The  $(n+1)$  column vectors are chosen to be  $v_0, v_1, \dots, v_n$  written as column vectors.

#### Algorithm KGHL:

The following algorithm converts a Karnin-Greene-Hellman scheme to a Blakely scheme.

1. Choose  $S$  to be the point given by  $U$ .
2. The hyperplanes  $H_1, H_2, \dots, H_n$  are chosen to be the hyperplanes whose coefficient vectors are given by  $v_1, v_2, \dots, v_n$ .

#### Algorithm BMSH:

This algorithm converts a Bloom's scheme to a Shamir's scheme.

1. Define  $v_0 = (1, 0, \dots, 0)$
2. Define the vectors  $v_i$  for  $1 \leq i \leq n$  as  

$$v_i = (1, b_i, b_i^2, \dots, b_i^{t-1}).$$
 Note that by lemma 7 any  $t$  of the  $v_i$ 's for  $i=0, 2, \dots, n$  are linearly independent.
3. Let  $a = (a_0, a_1, \dots, a_{t-1})$  and define  $L$  as,  

$$L(v) = a \cdot v \text{ for } v \text{ belonging to } v, \text{ where } a \cdot v \text{ is the dot product of } a \text{ and } v.$$

#### Algorithm GSDL:

This algorithm specializes a generalized linear scheme to a Blakely scheme.

1. Choose  $d = 0$ .
2. Choose the linear functional  $f$  to be a projection on one of the coordinate axis.

In view of these algorithms the various threshold schemes are specializations of the Generalized linear threshold scheme and they subsume Shamir's scheme.

#### 6. HIERARCHICAL THRESHOLD SCHEME:

In many applications there is a hierarchy among the trustees to whom the shadows are distributed for safeguarding the secret, and there is a need to create shadows of different potency. For example in a company where there are two levels of guards like senior and junior executives, it may be required that the threshold to obtain the secret be strictly smaller for senior executives compared to the threshold required of junior executives. In defense applications, this requirement may be even more crucial when there are different levels of commands and it is required that the lower

the level of command the higher the number of officers required to reveal the secret. These applications require a threshold scheme which provides shadows at different levels and the threshold is dependent on the level, such a scheme will be called a hierarchical threshold scheme.

An obvious solution to obtain a hierarchical threshold scheme seems to be to adopt an ordinary threshold scheme to the purpose by providing multiple shadows as shadows at higher levels. However this approach has drawbacks such as even though less shadows are required to reveal the secret the computation required in the process is not reduced, the shadows at different levels have to be physically different and interpreted differently, and a full range of threshold values is not available. A natural solution to these problems is offered by a generalized linear scheme.

A hierarchical threshold scheme is obtained from generalized linear threshold scheme as follows. The basic idea is to use linear varieties of different dimensions to conceal the secrets, at different levels.

Assume that  $V$  is a  $t$  dimensional vector space and  $f$  is a linear functional which is used to reveal the secret concealed by a linear subvariety  $S$ . The variety  $S$  is such that function  $f$  has a unique value, on  $S$ . There are several choices for  $S$  and the maximum dimension for  $S$  is  $(t-1)$ . The threshold is the difference between  $t$  and the dimension of  $S$ . Choose a sequence  $S_i$ ,  $0 \leq i \leq t-1$ , of linear subvarieties such that  $\dim(S_i) = i$  and  $f(S_i)$  is a single value giving the secret. The shadows at level  $i$  are generated with respect to  $S_i$  as described in the construction of generalized linear threshold scheme. The threshold at level  $i$  is then  $t-i$ . Thus we get a hierarchical scheme providing  $t$  different levels of shadows. At level  $i$ ,  $t-i$  linear equations have to be solved to obtain the secret, thus the computation to reveal the secret is proportional to the threshold.

## 7. CONCLUSIONS

This paper shows that various linear threshold schemes are closely related and they are all founded on the same principles. A generalized linear threshold scheme nicely crystalizes the basic principles of linear threshold schemes. The generalized threshold scheme has the flexibility which allows a chain of linear varieties to be used to conceal a secret. This property provides a hierarchical threshold scheme. The linear threshold schemes have fallen back on Shamir's method for a concrete implementation - however the same method cannot be used to implement a generalized threshold scheme as a hierarchical scheme. At this time we do not know of any efficient implementation of a hierarchical threshold scheme and the problem needs to be investigated further.

There are other possible generalizations of linear threshold schemes. The linear functional used to reveal the secret may be replaced by a fractional linear transformation (this is the case with Blakeley's projective threshold scheme) also the shadows may be chosen to be lower dimensional linear varieties instead of linear

hyperplanes. It is not clear that these generalizations would give any better threshold schemes.

#### ACKNOWLEDGEMENTS:

I thank Professor Blakely for the stimulating and valuable discussions on the subject of this paper. I thank Dr. Lakshmivarahan for initiating my interest in the subject. I thank Dr. Robert Roberts for suggesting some improvements in the manuscript.

#### REFERENCES:

- [Blakely 79] Blakely G.R., "Safeguarding Cryptographic Keys", Proceeding of the National Computer Conference, 1979, AFIPS Conference Proceedings, Vol. 48 (1979), pp. 313-317.
- [Blakely 81] Blakely G.R. and Laif Swanson, "Security Proofs for Information Protection Systems", Proceedings of the 1981 symposium on Security and Privacy, IEEE Computer Society, 1981, pp. 75-88.
- [Bloom 81] Bloom J.R., "A Note on Superfast Threshold Schemes", Preprint Texas A&M University, Department of Mathematics (1981).
- [Diffie 76] Diffie W. and Hellman, M.E., "New Directions in Cryptography", IEEE Trans. Inform. Theory, Vol. IT-22, No. 6, November 1976, pp. 644-654.
- [Karnin 83] Karnin, Greene and Hellman, "On Secret Sharing Systems", IEEE Transactions on Information Theory, Vol. IT-29, (1983), pp. 35-41.
- [Kuiper 65] Kuiper, "Linear Algebra and Geometry", North-Holland Publishing Company-Amsterdam, Second Edition, 1965.
- [Rivest 78] Rivest R.L., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Vol. 21, 1978, pp. 120-126.
- [Shamir 79] Shamir A., "How to Share a Secret", Communications of the ACM, Vol. 22, No. 11, Nov. 1979, pp. 612-613.
- [Shannon] Shannon C.E., "Communication Theory of Secrecy Systems", Bell System Technical Journal, 1948, pp. 656-715.

## SECURITY OF RAMP SCHEMES

G. R. Blakley and Catherine Meadows

Department of Mathematics  
Texas A&M University  
College Station, Texas 77843-3368

### 1. OVERVIEW.

A  $k$  out of  $n$  p/s/r process [AS81] is a very efficient way to convey information ( $k$  words suffice to reclaim  $k$  words). But it provides virtually no cryptographic security for the information it deals with.

A  $k$  out of  $n$  threshold scheme [DE81, p. 179-187] is very inefficient as a conveyor of information ( $k$  words are necessary to reclaim 1 word). But the linear threshold schemes provide Shannon perfect security [BL81a] up to threshold  $k$ . Examples of linear threshold schemes are Blakley projective [BL79], Blakley affine [BL73], Shamir [SH79], Bloom [BL81b], McEliece/Sarwate [MC81], some versions of Asmuth/Bloom [AS83] and some versions of Karnin/Greene/Hellman [KE83]. In addition to the linear threshold schemes there are the threshold schemes due to Davida/DeMillo/Lipton [DA80], some Asmuth/Bloom schemes, and some Karnin/Green/Hellman schemes.

For many practical purposes, Shannon perfect security is too much security if it is bought with  $k$ -fold (or more) bandwidth expansion. A magazine wanting to use a 4 out of 6 threshold scheme to store a mailing list occupying 12 rolls of magnetic tape might balk at the need to write, store and manipulate 72 rolls of mag tape to gain Shannon perfect security against opponents whose cryptanalytic expertise is unimpressive. But it might be willing to write, store and handle 24 rolls to get a specified -- more modest -- level of security, the reasoning being much the same as what leads people to put locks on glass doors. You balance level of security against the amenities which less security provides, in an environment in which the opponents are viewed as troublesome but not too threatening.



We will follow a suggestion of Bloom's [BL81b] and explore the properties of various versions of what we will call a "k out of n to yield d ramp scheme" (or, more briefly, a (d,k,n) ramp scheme). Figures 1.1, 1.2 and 1.3 will make clear why we chose the ramp terminology for the generalization of the notion of threshold scheme.

One of the most important types of ramp scheme, the linear ramp scheme does the following. It takes d pieces of input information (i.e. members of a finite field F). From these d inputs (and using k - d other predetermined types of inputs, perhaps some of them random) it produces n outputs in such a fashion that the d inputs can easily be reconstructed from any k outputs. But there is a predetermined level of uncertainty (perhaps the level is zero, and there is an absolute upper bound dependent on j) regarding the inputs if only j outputs are known -- given that j < k. It should be obvious from the description above that the assumption

$$1 \leq d \leq k \leq n$$

is implicit in this definition.

The magazine mentioned above could use a (3,4,6) ramp scheme to turn its 12 input rolls of mag tape into 6 boxes (each containing four output rolls). This ramp scheme would have the property that it is easy to get the contents of all 12 input rolls back from any 4 boxes of four output rolls. A competitor of the magazine who gained access to only 3 of these boxes of four tapes each would have some knowledge of the contents of the 12 original mag tape rolls, but likely not enough to be useful.

The basic security consideration in a linear k out of n threshold scheme is all-or-none, i.e. Shannon perfect security [BL81b; SH79]. For every word w belonging to the field F we have

$$\text{Probability (w is the word conveyed by the scheme} \mid \begin{array}{l} \text{given} \\ \text{that } k-1 \text{ (or fewer) shadows are known)} \end{array})$$

$$= \text{Probability (w is the word conveyed by the scheme)}$$

In other words, no amount of knowledge of shadows [BL79] (coded words) below the threshold level k enables a Bayesian opponent [K081, p. 31] to modify an a priori guess regarding what information the scheme conveys.

A k out of n threshold scheme is the extreme (1,k,n) case of the notion of ramp scheme. See Figure 1.2 below for a description

of a linear  $(1,k,n)$  ramp scheme. A  $k$  out of  $n$  p/s/r process is the opposite extreme, the  $(k,k,n)$  case of the notion of ramp scheme. See Figure 1.3 below for a description of a linear  $(k,k,n)$  ramp scheme). Here there is a small measure of security. If you intercept only  $k-1$  shadows you can know at most  $(k-1)/100k$  per cent of the  $k$  pieces of information that were to be conveyed. And you may know less than that, depending on circumstances. We will address this point more fully below.

The basic security consideration in a  $(d,k,n)$  ramp scheme is Shannon relative security. This generalization of the notion of Shannon perfect security goes as follows. Consider the  $d$  dimensional vector space  $T$  consisting of all lists

$$(f(1), f(2), \dots, f(d))$$

of  $d$  words (i.e. members of the finite field  $F$  in question). Somebody who knows how the linear ramp scheme in question has been designed and implemented can do no better than the following. Given knowledge of  $z$  shadows of the information there is an affine subspace  $U$  of  $T$ . The dimensionality  $\dim(U)$  of this affine subspace  $U$  is

$$\dim(U) = \min\{d, \max\{0, k-z\}\}$$

(see Figure 1). The subspace  $U$  has the property that for every list

$$\xi = (\xi(1), \xi(2), \dots, \xi(d))$$

of elements of the field  $F$  in question we have

Probability (the list  $\xi$  to be conveyed does not belong to  $U$ ) = 0,

Probability (the list  $\xi$  to be conveyed is equal to  $w \in U$  | given the knowledge of  $z$  intercepted shadows)

$$= \frac{\text{Probability (the list } \xi \text{ to be conveyed is equal to } w \in U)}{\text{Probability (the list } \xi \text{ belongs to } U)}$$

In brief, a Bayesian opponent in possession of  $z$  shadows from a  $(d,k,n)$  linear ramp scheme now knows that the desired list  $\xi$  belongs to  $U$ . This is a considerable increase over the amount of information he had at the outset, before he knew any shadows. But, as to where it is within  $U$ , he knows no more than he did before he had acquired any shadows. Thus suppose that  $1 \leq d \leq k \leq n$ . With  $z$  shadows

available, an opponent knows of a subspace  $U$  whose dimension is given in the Figure 1.1 below.

The concept of linear ramp scheme can also be extended to more general (nonlinear) ramp schemes (such as some versions of the Asmuth/Bloom ramp scheme) by associating a subset  $U$  of  $T$  to each set of  $z$  shadows, where  $T$  is merely a set of lists of  $d$  words instead of a vector space. Of course in this case we cannot put dimensionality requirements on  $U$ , since  $U$  will in general not be a linear space. We can, however put degree-of-freedom requirements on  $U$ , namely, that if  $U$  the subset of  $T$  corresponding to  $k+d+s$  shadows then knowledge of  $U$  leaves us with exactly  $d-s$  degrees of freedom. In other words, knowledge of  $U$  should not give us any information about any  $d-s$  member sublist of  $\xi = (\xi(1), \xi(2), \dots, \xi(d))$  but knowledge of  $U$  and any  $d$ -member sublist should give us knowledge of the whole list.

We will follow, to some extent, a recent view of threshold schemes due to S. Kothari [K085]. In addition to its unifying properties, his formulation neatly uncouples the probabilistic considerations from the algebra. This makes it possible for him to give simple elegant proofs of Shannon perfect security for many heretofore seemingly different threshold schemes. Also, he makes explicit the notion that Shamir's [SH79] threshold scheme is a special case of Bloom's [BL81b]. We wish to point out that the converse statement also holds, in a sense. The first mention of this converse to Kothari's observation can be found in [KA83]. Thus it might be more appropriate to speak of a Shamir/Bloom scheme--or of the Bloom approach to a Shamir threshold scheme--henceforward, rather than of separate threshold schemes. Kothari also shows that the Bloom threshold scheme [K085] is dual to the Blakley affine [BL83] geometric threshold scheme. So all the known Shannon perfectly secure threshold schemes are linear algebraic, and are to all mathematical intents and purposes identical. A corollary of this is that there are rigid [BL83] Blakley schemes and nonrigid versions of the other schemes.

Since the theory of threshold schemes and ramp schemes seems to be maturing, we have collected all the papers touching it known to us in the references at the end of this paper. It is worth noting that Chaum also enunciated ideas [CH79; CH82] along the lines of threshold schemes and suggested implementations making use of cryptosystems.

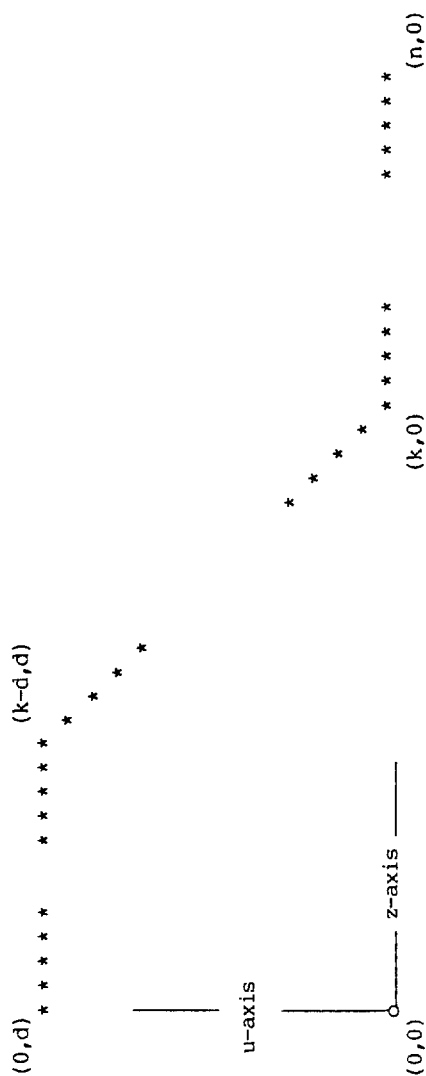


Figure 1.1

Graph of number  $u$  of degrees of uncertainty (i.e. of the dimension  $u$  of the subspace  $U$ ) versus number  $z$  of known shadows in the general  $(d,k,n)$  linear ramp scheme. The ramp falls at a 45 degree angle from  $(z,u) = (k-d,d)$  to  $(z,u) = (k,0)$ .

$z$	$\min\{d, \max\{0, k-z\}\}$
0	$d$
1	$d$
2	$d$
...	$d$
$k-d-2$	$d$
$k-d-1$	$d$
$k-d$	$d$
$k-d+1$	$d-1$
$k-d+2$	$d-2$
...	...
$k-2$	2
$k-1$	1
$k$	0
$k+1$	0
$k+2$	0
...	0
$n-2$	0
$n-1$	0
$n$	0

z	$\min\{1, \max\{0, k-z\}\}$
0	1
1	1
2	1
...	1
k-2	1
k-1	1
k	0
k+1	0
k+2	0
...	0
n-2	0
n-1	0
n	0

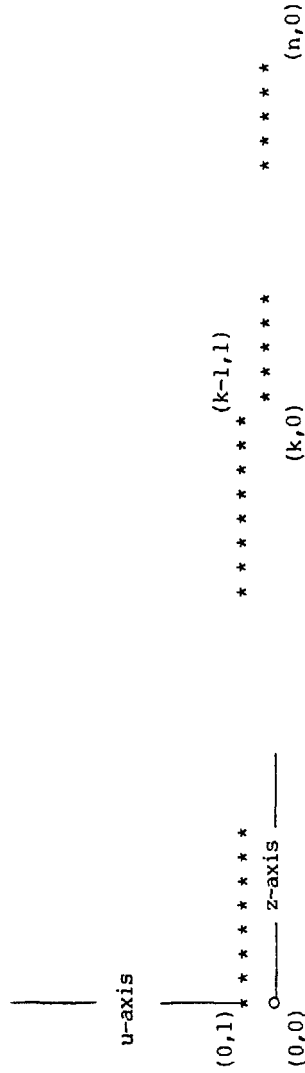


Figure 1.2

Graph of number  $u$  degrees of uncertainty (i.e. of the dimension  $u$  of the subspace  $u$ ) versus number  $z$  of known shadows in a  $(1,k,n)$  linear ramp scheme, i.e. a  $k$  out of  $n$  linear threshold scheme. The ramp falls at a  $45^\circ$  angle from  $(z,u) = (k-1,1)$  to  $(z,u) = (k,0)$ .

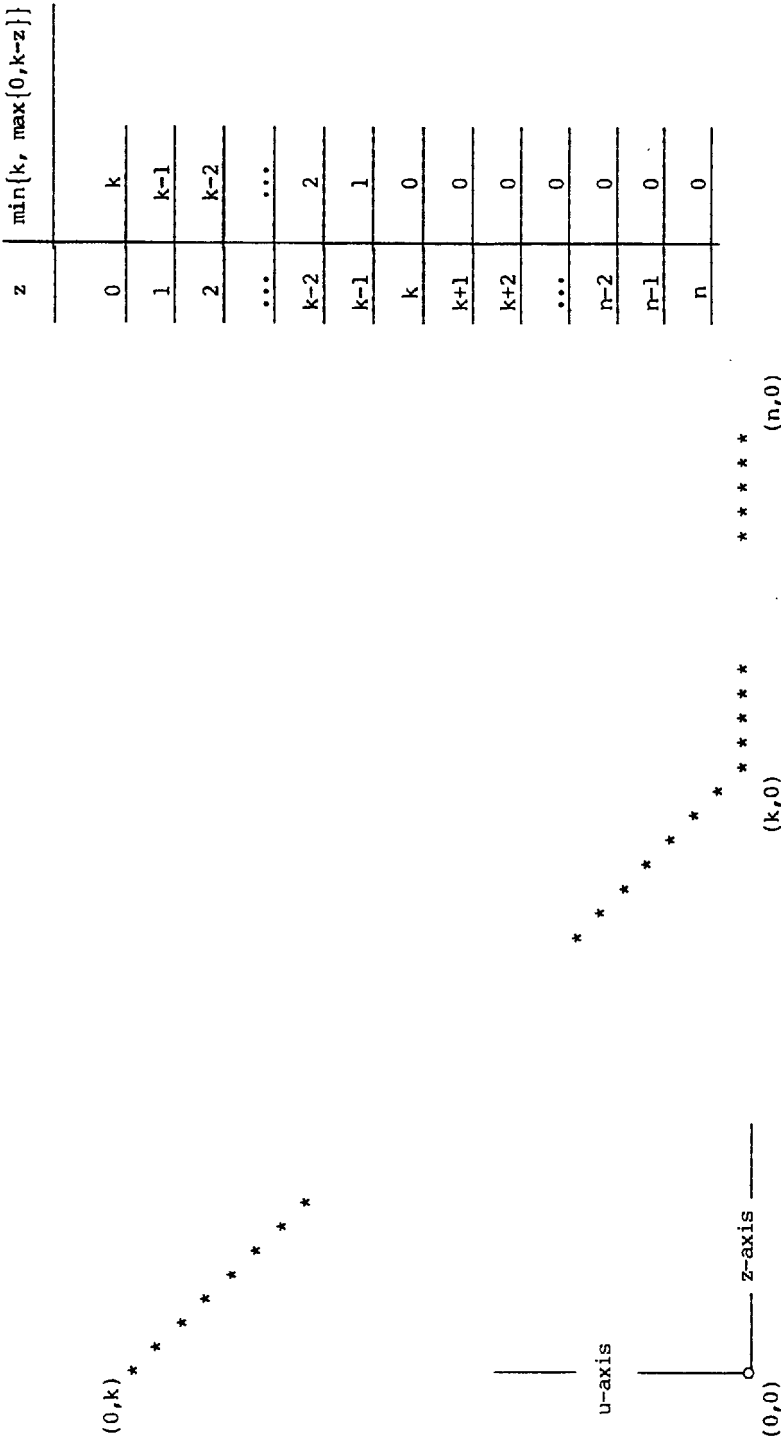


Figure 1.3

Graph of number  $u$  degrees of uncertainty (i.e. of the dimension  $u$  of the subspace  $U$ ) versus number  $z$  of known shadows in a  $(k, k, n)$  linear ramp scheme, i.e. a  $k$  out of  $n$  linear  $p/s/r$  process. The ramp falls at a  $45^\circ$  angle from  $(z, u) = (0, k)$  to  $(z, u) = (k, 0)$ .

## 2. GENERALIZED RAMP SCHEMES

In this section we develop a general definition of ramp scheme. This definition is actually more general than needed for the examples in this paper, but we state it in as much generality as possible in order that it can be made to fit any future examples of ramp schemes.

(2.1) Definition. A  $(d, k, n)$  ramp scheme is defined as follows. We start with a concealing set  $V$  and a key set  $W$  such that

$$\frac{\log |V|}{\log |W|} \approx \frac{k}{d}$$

where  $| \cdot |$  denotes cardinality. Let  $\pi$  be a surjective map from  $V$  to  $W$ , that is let  $\pi$  be a map such that for every  $w \in W$ , there is at least one  $v \in V$  such that  $\pi(v) = w$ . We will call  $\pi$  the revealing map. Given a key element  $w$  in  $W$  we choose point  $y$  in  $\pi^{-1}(w)$  called the concealing point. To this point  $y$  we associate a set of  $n$  shadows

$$\{H(1), H(2), \dots, H(n)\}$$

where each shadow  $H(i)$  is a subset of  $V$  such that

- a) The intersection of any  $k$  shadows is  $\{y\}$ .
- b) There exists an integer  $l$  dependent upon  $d$  and  $k$  such that
  - i)  $1 \leq l \leq k$
  - ii) The restriction of  $\pi$  to the intersection of any  $l$  shadows is surjective.
  - iii) Knowledge about  $w = \pi(y)$  increases in some regular way with knowledge of each shadow after  $l$  shadows.

As an example of what we mean by the last part of this definition, suppose that  $W$  is a vector space of dimension  $s$  over a finite field. We could require that, if  $H$  is the intersection of  $l+i < k$  shadows, then  $\pi(H)$  is a vector space of dimension  $s-i$ .

In the threshold scheme case  $(1, k, n)$  we can require a scheme to be Shannon perfectly secure. We define a threshold scheme to be Shannon perfectly secure if it satisfies the following criterion.

Whenever the intersection  $H$  of less than  $k$  shadows is known, then the probability that the image of  $x$  under  $\pi$  is  $w$  is equal to the a priori probability of  $w$  (in other words,  $p(\pi^{-1}(w)|H) = p(w)$ ). Since some probabilities go to zero in the general ramp scheme case, and the remaining ones usually increase, Shannon perfect security is of course impossible. However, we can still require that no probability which remains positive increases any faster than any other which remains positive, i.e. that the ratios of the remaining positive probabilities remain the same. If this is not possible we can at least require that the ratios do not vary too much from the original. The following definition makes these ideas precise.

(2.2) Definition. A  $(d,k,n)$  ramp scheme  $R$  is Shannon relatively secure if, whenever the intersection  $H$  of less than  $k$  shadows is known, then

$$(2.3) \quad \frac{p(\pi^{-1}(w)|H)}{p(\pi^{-1}(w^*)|H)} = \frac{p(w)}{p(w^*)}$$

for every pair of elements  $w$  and  $w^*$  in  $\pi(H)$ . A  $(d,k,n)$  ramp scheme  $R$  is Shannon  $t$ -relatively secure if, whenever the intersection  $H$  of less than  $k$  shadows is known, then

$$(2.4) \quad \frac{p(w)}{(1+t)p(w^*)} < \frac{p(\pi^{-1}(w)|H)}{p(\pi^{-1}(w^*)|H)} < \frac{(1+t)p(w)}{p(w^*)}.$$

We say that a ramp scheme is  $t$ -relatively secure with knowledge of  $r$  shadows if the above inequality holds whenever  $H$  is the intersection of  $r$  shadows.

In the following lemma we show that the definition of Shannon relative security arises naturally from Shannon perfect security.

(2.5) Lemma. Let  $R$  be a  $(d,k,n)$  ramp scheme. Then  $R$  is Shannon relatively secure if and only if whenever the intersection  $H$  of less than  $k$  shadows is known, then

$$p(\pi^{-1}(w)|H) = p(w|\pi(H)).$$

In particular, a threshold scheme is Shannon relatively secure if and only if it is Shannon perfectly secure, and a Shannon relatively secure ramp scheme is Shannon perfectly secure up to and including



knowledge of  $k - d$  shadows.

Proof: Let  $H$  be the intersection of no more than  $k$  shadows. If

$$p(\pi^{-1}(w) | H) = p(w | \pi(H))$$

then for all  $w \in \pi(H)$

$$\begin{aligned} p(\pi^{-1}(w) | H) &= p(w) / w(H) \\ &= p(\{w\} \cap \pi(H)) / p(\pi(H)) \\ &= p(w) / p(\pi(H)). \end{aligned}$$

Thus, for all  $w$  and  $w^*$  in  $\pi(H)$  we have

$$\frac{p(\pi^{-1}(w) | H)}{p(\pi^{-1}(w^*) | H)} = \frac{p(w) / p(\pi(H))}{p(w^*) / p(\pi(H))} = \frac{p(w)}{p(w^*)}$$

and so  $R$  is Shannon relatively secure.

Conversely, suppose that  $R$  is Shannon relatively secure.

If  $w \in \pi(H)$  then both  $p(\pi^{-1}(w) | H)$  and  $p(w | \pi(H))$  are zero. Now let  $\pi(H) = \{w(0), w(1), \dots, w(m)\}$ . Then

$$p(\pi^{-1}(w(0)) | H) / p(\pi^{-1}(w(i)) | H) = p(w(0)) / p(w(i))$$

for  $1 \leq i \leq m$ . Moreover

$$p(\pi^{-1}(w(0)) | H) + \dots + p(\pi^{-1}(w(m)) | H) = 1.$$

This gives us a nondegenerate system of  $m + 1$  linear equations in  $m + 1$  unknowns  $p(\pi^{-1}(w(0)) | H)$  through  $p(\pi^{-1}(w(m)) | H)$  for which

$$\begin{aligned} p(\pi^{-1}(w(0)) | H) &= p(w(0) | \pi(H)), \\ &\vdots \\ p(\pi^{-1}(w(m)) | H) &= p(w(m) | \pi(H)) \end{aligned}$$

is the unique solution.

The last statement of the lemma follows from the fact that

$\pi(H) = W$  when no more than  $k - d$  shadows are known.

The major advantage of the above definition of ramp schemes is that, as in the case of Kothari's [K085] definition of linear threshold scheme, it allows us to characterize Shannon relative security solely in terms of the cardinalities of  $\pi^{-1}(w)$  and  $\pi^{-1}(w) \cap H$ .

(2.6) Lemma. Let  $R$  be a  $(d, k, n)$  ramp scheme. Then  $R$  is Shannon relatively secure if and only if, whenever the intersection  $H$  of fewer than  $k$  shadows is known, the equality

$$(2.7) \quad \frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} = \frac{|\pi^{-1}(w)|}{|\pi^{-1}(w^*)|}$$

holds for all  $w, w^*$  in  $\pi(H)$ .  $R$  is Shannon  $t$ -relatively secure if and only if, whenever the intersection  $H$  of fewer than  $k$  shadows is known, the inequalities

$$(2.8) \quad \frac{|\pi^{-1}(w)|}{(1+t) |\pi^{-1}(w^*)|} < \frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} < \frac{(1+t) |\pi^{-1}(w)|}{|\pi^{-1}(w^*)|}$$

hold for all  $w, w^*$  in  $\pi(H)$ .

Proof.  $R$  is Shannon relatively secure if and only if

$$p(w|H)/p(w^*|H) = p(w)/p(w^*).$$

But

$$p(w|H) = p(\pi^{-1}(w) \cap H)/p(H).$$

Since the point  $y$  in  $\pi^{-1}(w)$  is chosen at random the distribution on  $\pi^{-1}(w)$  is uniform and so

$$p(\pi^{-1}(w) \cap H) = \frac{p(w) |\pi^{-1}(w) \cap H|}{|\pi^{-1}(w)|}$$

Thus equation (2.3) in the definition of Shannon relative security becomes

$$\frac{p(w)}{p(w^*)} \frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} \frac{|\pi^{-1}(w^*)|}{|\pi^{-1}(w)|} = \frac{p(w)}{p(w^*)}$$

which reduces to equation (2.7). Similarly, inequality (2.4) in the definition of Shannon  $t$ -relative security becomes

$$\frac{p(w)}{(1+t)p(w^*)} < \frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} \frac{|\pi^{-1}(w^*)|}{|\pi^{-1}(w)|} \frac{p(w)}{p(w^*)} < \frac{(1+t)p(w)}{p(w^*)}$$

which reduces to inequality (2.8). □

In the remaining sections of this paper we examine several examples of  $(d,k,n)$  ramp schemes, along with their security proofs.

### 3. RIGID LINEAR SCHEMES

In this section we develop a general linear ramp scheme within the general framework set forth by Kothari in [KO85]. Namely, let  $V$  be a vector space of dimension  $k$  over a finite field  $F$ , let  $y$  be a point of  $V$  hiding the key vector in some way, and let the  $H(i)$ s be hyperplanes in general position with respect to  $y$ . (This is actually less general than Kothari's scheme, which also includes projective and affine spaces.) We let  $W$  be  $F^d$  and we let the map  $\pi: V \rightarrow W$  be a linear transformation. It turns out that if the  $H(i)$ s are oriented so that the intersection of any  $l$  of them with a certain translation of the kernel of  $\pi$  is a linear variety of dimension  $\min(k-l-d, 0)$  then any such scheme is Shannon relatively secure. We make this requirement precise below.

(3.1) Definition. We define a  $(d,k,n)$  linear ramp scheme in the following way. Let  $F$  be a finite field, let  $V = F^k$  and let  $W = F^d$ . Let  $\pi: F^k \rightarrow F^d$  be a linear transformation. Choose hyperplanes  $T(1), T(2), \dots, T(d)$  through the origin such that  $T(1) \dots T(d)$  is the kernel of  $\pi$ . For  $1 \leq i \leq d$ , let  $t(i)$  be the vector such that

$$T(i) = \{x \in F^k \mid t(i) \cdot x = 0\}.$$

Next choose hyperplanes through the origin  $S(1), \dots, S(n)$  such that the set

$$\{T(1), \dots, T(d), S(1), \dots, S(n)\}$$

is in general position, with respect to the origin, that is, such that the intersection of any  $k$  members of the set is the origin. For  $1 \leq i \leq n$  let  $s(i)$  be the vector such that

$$S(i) = \{x \in F^k \mid s(i) \cdot x = 0\}.$$

(Thus the set of vectors  $\{t(1), \dots, t(d), s(1), \dots, s(n)\}$  is in general position.) For a given  $w$  in  $F^d$  we choose shadows by picking a point  $y$  at random in  $\pi^{-1}(w)$  and then picking field elements  $c(1)$  through  $c(n)$  such that the intersection of any  $k$  of the hyperplanes

$$H(i) = \{x \in F^k \mid x \cdot s(i) = c(i)\}$$

is  $\{y\}$ . Since the hyperplanes  $S(i)$  are in general position, this can be done by choosing  $c(i)$  such that  $y$  is an element of each  $H(i)$ . Since  $y$  is also an element of each  $T(i)$ , the set

$$\{T(1), \dots, T(d), H(1), \dots, H(n)\}$$

is also in general position with respect to  $y$ .

(3.2) Proposition. The linear scheme is a  $(d, k, n)$  ramp scheme with  $k = d - s$ . Moreover, if  $k - d + s$  shadows are known, then all that is known about the key element  $w$  is that it lies in a vector subspace of  $W$  of dimension  $d - i$ .

Proof: That linear schemes satisfy condition (a) of Definition (2.1) is clear from their definition.

In order to prove the rest of the proposition, let  $H$  be the intersection of  $r$  shadows, where  $1 \leq r \leq n$ . Thus

$$H = H(i_1) \cap \dots \cap H(i_r).$$

Let

$$S = S(i_1) \cap \dots \cap S(i_r).$$

Then  $H = S + a$ , where  $a \in F^k$ . It follows that  $\dim(\pi(H)) = \dim(\pi(S))$ , and so it remains to show that  $\pi(S) = F^d$  when  $r \leq d - k$  and  $\dim(\pi(S)) = d - s$  when  $r = d - k + s$ .

Since the  $S(i)$ s and the  $T(i)$ s are in general position, we have

$$\begin{aligned}
 \dim(S \cap T) &= \min(0, \dim(S) + \dim(T) - k) \\
 &= \min(0, k-r+k-d-k) \\
 &= \min(0, k-(d+r)).
 \end{aligned}$$

It follows that, if  $r = k-d-s$  where  $0 \leq s < k-d$ , then

$$\begin{aligned}
 \dim(\pi(S)) &= \dim(S) - \dim(S \cap T) \\
 &= k-r-(k-(d+r)) \\
 &= d
 \end{aligned}$$

Thus  $\pi(S)$ , and hence  $\pi(H)$ , is all of  $F^d$ . If  $r = k-d+s$ , where  $1 \leq s \leq d$ , then

$$\begin{aligned}
 \dim(\pi(S)) &= \dim(S) - \dim(S \cap T) \\
 &= k-r-0 \\
 &= k-(k-d+s) \\
 &= d-s.
 \end{aligned}$$

Hence  $\dim(\pi(H)) = d-s$ . □

(3.3) Remark. We note that if the subspaces  $S(i), \dots, S(n)$  are fixed beforehand and made known to the general public, no information about the key element  $w$  is given away. Moreover, we are saved the trouble of calculating the orientations of the  $H(i)$ s anew each time, and economy is gained since each shadow holder only has to guard a single field element instead of the equation of a hyperplane. It follows that any efficient ramp scheme will have this property. However, as we see, this property can be built into any linear ramp scheme.

Schemes in which the  $S(i)$ s are fixed beforehand are known as rigid schemes since the choice of the shadows  $H(i)$  is fixed by the choice of the point  $y \in \pi^{-1}(w)$ .

(3.4) Remark: If we think of the key element  $w$  as a single random key, then the above construction of rigid linear ramp scheme suffices. However, if we think of  $w$  as a vector of  $d$  keys, or as a word that could possibly be deduced if we knew a small part of it, we need to put further conditions on the scheme. For example, we want to avoid such instances as

$$\pi(H(i_1) \cap \dots \cap H(i_{k-d+s})) = \{w_1\} \times \dots \times \{w_s\} \times F^{d-s}$$

since this would give away  $s$  of the keys. In other words, if

$$\rho_j: F^d \rightarrow F$$

is the projection defined by  $\rho((f_1, \dots, f_d)) = f_j$  we never want  $\rho_j(\pi(H))$  to be a single point of  $H$  if  $H$  is the intersection of less than  $k$  shadows. But since  $\pi(H)$  is a translation of  $\pi(S)$ , where  $S$  is the intersection of the  $S(i)$ s corresponding to the  $H(i)$ s whose intersection is  $H$ , it is enough to require that  $\rho_j(\pi(S)) \neq \{0\}$ . This can be done easily by first setting the subspaces  $T(j)$  equal to  $\ker(\rho_j \circ \pi)$ . Now suppose in such a case that  $\rho_j(\pi(S)) = \{0\}$ . Then  $S \subseteq \ker(\rho_j \circ \pi) = T(j)$ . But this contradicts the fact that the  $S(i)$ s will have been chosen such that the  $S(i)$ s and the  $T(j)$ s are in general position.

(3.5) Theorem. The  $(d, k, n)$  linear ramp scheme is Shannon relatively secure.

Proof: By Lemma (2.6) it is enough to show that given  $H$  the intersection of  $l$  shadows, then

$$\frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} = \frac{|\pi^{-1}(w)|}{|\pi^{-1}(w^*)|}$$

for all  $w$  and  $w^*$  in  $\pi(H)$ . Since  $\pi$  is a linear transformation, we know that

$$|\pi^{-1}(w)| = |\pi^{-1}(w^*)|$$

for all  $w$  in  $F^d$ , so it is enough to show that

$$|\pi^{-1}(w) \cap H| = |\pi^{-1}(w^*) \cap H|$$

for all  $w$  and  $w^*$  in  $\pi(H)$ .

Since  $\pi^{-1}(w)$  is a translation of the kernel of  $\pi$ ,  $\pi^{-1}(w)$  and  $H$  are translations of two vector subspaces in general position. Thus, if the sum of their dimensions is less than  $k$  (that is, if  $H$  is the intersection of more than  $k-d$  shadows) their intersection is either empty or a single point, and  $|\pi^{-1}(w) \cap H| = 1$  whenever

$w \in \pi(H)$ . If the sum of the dimensions of  $\pi^{-1}(w)$  and  $H$  is greater than  $k$ , then

$$\dim(\pi^{-1}(w) \cap H) = \dim(\pi^{-1}(w)) + \dim(H) - k$$

for all  $w$ , so  $|\pi^{-1}(w) \cap H|$  is the same for all  $w$ .

Next we look at some examples of ramp schemes.

#### A. Blakley Scheme

The Blakley scheme is the scheme of Definition (3.1) with  $\pi$  taken to be a projection to a  $d$ -dimensional subspace of  $F^k$  such that kernel of  $\pi$  satisfies the conditions of Definition (3.1). This is essentially a rigid version of the Blakley scheme described in [BL79].

#### B. Bloom Scheme

In this scheme [BL81b] we let  $V$  be  $(F^k)^*$ , the space of linear functionals [HO71, p. 97] from  $F^k$  to  $F$ . (Of course  $(F^k)^*$  is isomorphic to  $F^k$ .) Let  $t_1, \dots, t_d$  be linearly independent vectors in  $F^k$ . The map  $\pi: (F^k)^* \rightarrow F^d$  is given by

$$\pi(L) = (L(t_1), \dots, L(t_d)).$$

Choose vectors  $\{s_1, \dots, s_n\}$  in  $F^k$  such that the set

$$\{t_1, \dots, t_d, s_1, \dots, s_n\}$$

is in general position. let

$$S(i) = \{L \in (F^k)^* \mid L(s_i) = 0\}$$

and let

$$T(i) = \{L \in (F^k)^* \mid L(t_i) = 0\}.$$

Then if we let  $T = T(1) \cap \dots \cap T(d)$ , we have  $T = \ker(\pi)$ . Let  $w$  be a point in  $F$ . Pick a linear functional  $G$  at random in  $\pi^{-1}(w)$ . The shadows  $H(i)$  associated to  $w$  are

$$H(i) = \{L \in (F^k)^* \mid L(s_i) = G(s_i)\}.$$

Clearly  $H(i)$  is a translation of  $S(i)$ , and so the Bloom scheme satisfies all the criteria for a linear ramp scheme. Moreover, it follows from the way we have defined the  $T(i)$ s that the Bloom scheme clearly satisfies the criteria of Remark (3.3).

### C. Shamir Scheme

The Shamir threshold scheme [SH79] is defined as follows. Let  $f(x)$  be a polynomial of degree  $k-1$  in  $F[x]$ , where  $F$  is a finite field. Choose elements  $c_1, b_1, \dots, b_n$ . Let  $f(c_1)$  be the key and let  $f(b_1)$  through  $f(b_n)$ . If we know  $k$  shadows then we can use Lagrange interpolation to find  $f(x)$  and hence the key  $f(c_1)$ . In [K084] Kothari points out that the Shamir scheme is a special case of Bloom's scheme. Suppose that

$$f(x) = a_0 + a_1x + \dots + a_nx^{k-1}.$$

We let the linear functional  $L$  we are hiding be defined by

$$L(v) = (a_0, a_1, \dots, a_n) \cdot v,$$

we let the  $n$  vectors  $s_i$  in general position be

$$s_i = (1, b_i, (b_i)^2, \dots, (b_i)^{k-1})$$

and let  $t_1$  be  $(1, b_0, (b_0)^2, \dots, (b_0)^{k-1})$ . The key is  $L \cdot v_1$ . This is clearly equivalent to Shamir's scheme. Moreover, since these vectors (which make up the Vandermonde matrix) are usually the ones chosen for the Bloom scheme anyway this means that Bloom's and Shamir's schemes are essentially equivalent. We point out that we can make Shamir's scheme into a ramp scheme by letting the key element be the vector  $(f(c_1), \dots, f(c_d))$  where  $c_1, \dots, c_d$  are elements of  $F$ .

### D. Karnin/Greene/Hellman Scheme.

In this scheme [KA83] the vector space  $F^d$  is replaced by  $(F^e)^d$  and  $F^k$  by  $(F^e)^k$ , where  $e$  is a positive integer. Let  $A(1)$  through  $A(d)$  be  $k \cdot e$  by  $e$  matrices such that the  $k \cdot e$  by  $d \cdot e$  matrix formed by  $A(1)$  through  $A(d)$  is of maximal rank. Next choose matrices  $B(1)$  through  $B(n)$  such that any  $k$  member subset of



$$\{(A(1), \dots, A(d), B(1), \dots, B(n))\}$$

gives a  $k$ -e by  $k$ -e matrix of full rank. The map

$$\pi: (F^e)^k \rightarrow (F^e)^d$$

is given by

$$\pi(x) = (A(1) \cdot x, \dots, A(d) \cdot x).$$

Let

$$T(i) = \{u \in (F^e)^k \mid A(i) \cdot u = 0\}.$$

Let

$$S(j) = \{x \in (F^e)^k \mid B(j) \cdot x = 0\}.$$

Clearly the  $S(j)$ s and the  $T(j)$ s are in general position and the intersection of the  $T(i)$ s is the kernel of  $\pi$ . If  $w$  is a vector in  $(F^e)^d$  choose a random member  $u$  of  $\pi^{-1}(w)$ . We let the  $i$ th shadow associated with  $w$  be

$$H(i) = \{x \in (F^e)^k \mid B(i) \cdot x = B(i) \cdot u\}.$$

The conditions of the Karnin/Greene/Hellman scheme are similar to those of Definition (3.1) and similar security proofs may be obtained. In particular, the Karnin/Greene/Hellman scheme reduces to Bloom's scheme if  $e = 1$ . Simply replace  $F^k$  by  $(F^k)^*$  and the random vector  $u$  in  $\pi^{-1}(w)$  by  $u^* \in (F^k)^*$ , where  $u^*(x) = u \cdot x$  for every  $x \in F^k$ . Then  $\pi: F^k \rightarrow F^d$  becomes

$$\pi(u^*) = (u^*(A(1)), \dots, u^*(A(d))),$$

$S(i)$  becomes  $\{u^* \mid u^*(A(i)) = 0\}$  and so forth.

#### 4. FIELD SIZES

We have paid little attention to the field  $F$  underlying the vector spaces above. But with general  $(d, k, n)$  ramp scheme, as with its special case the  $k$  out of  $n$  threshold scheme and the  $k$  out of  $n$  p/s/r process it is necessary that the underlying field contain at least  $n$  members. The reason for this is that otherwise it is not possible to find the necessary hyperplanes (or points, as the case may

be) in general position as required by the formulation given in Section 3.

Actually the cardinality of the field can drop as low as  $n-2$  in some rather exceptional cases. There is, for example a Bloom 3 out of 6 p/s/r process (i.e. a Bloom (3,3,6) ramp scheme) whose underlying field is  $GF(4)$ . The reason for this is that the six vectors

[1,0,0]  
[0,1,0]  
[0,0,1]  
[1,1,1]  
[1,a,b]  
[1,b,a]

are in general position in  $GF(4)^3$ , where the members of  $GF(4)$  are 0,1,a and  $b = a^2$ .

But the  $n-2$  bound is seldom attained, and it is an open question [MA67] to characterize all the cases in which this happens.

For many purposes there is no advantage to be gained by using large field in building a threshold scheme (i.e. a (1,k,n) ramp scheme) if a smaller field would suffice. The latter, which could be implemented more cheaply and quickly, would provide as much security--Shannon perfect security--as the former.

But, when it comes to more general ramp schemes, there are many occasions on which use of a large underlying field might be desirable. Only Shannon relative security (or even merely Shannon  $t$ -relative security) is available to the user of a (d,k,n) ramp scheme when  $d \geq 2$ . So it might be desirable to have a large haystack of shadows in which to hide the message needle. Somebody who used  $GF(2^{32})$  as the underlying field for let us say, a (2,5,9) ramp scheme would have the consolation of knowing that an opponent who had obtained four shadows corresponding to a single test of two 32-bit words would still have nothing other than his a priori guess as to which of 4 billion possibilities was the correct value of the 64 bit string in question. The opponent would, of course, be better off for knowing the four shadows, having thereby eliminated more than 18 billion billion possibilities.

Contrast this state of affairs with 8 successive applications of a (2,5,9) ramp scheme over  $GF(16)$  to the same 64 bits of information. Each successive 8-bit substring would be narrowed down to one of 16 possibilities. The total possible number of values of the 64 bit string would again be some what over 4 billion to an opponent who had intercepted four shadows of everything. But, though

the opponents recovery problems in the two cases at hand thus look mathematically equivalent, they are not cryptographically equivalent. Suppose for example, that the original 64 bit string were eight latin letters ASCII coded, and each of the successive two-halbyte lists were the ASCII code for a single letter. The 16 possibilities for each symbol would be narrowed down to 1 or 2 by the requirement that it be one of only 26 bytes among all 256 possible bytes. The opponent would thus recover the message without machine assistance if the underlying field were  $GF(16)$ . No such cheap approach is available when  $GF(2^{32})$  is employed.

The question of infinite fields is a different matter, as noted in [BL83]. However the duality relationship between Blakley schemes and Shamir/Bloom schemes enables us to give a satisfactory solution to some of the problems raised in [BL83]. We can now produce Shannon perfectly secure rigid Blakley projective geometric  $(d,k,n)$  ramp schemes which amount to natural, and very efficient, generalizations of infinite one-time pads [BL83]. This development will be described in full elsewhere.

## 5. ASMUTH/BLOOM SCHEMES

In this section we look at an example of a ramp scheme based on the Asmuth/Bloom threshold scheme [AS83].

Choose prime numbers  $p(1)$  through  $p(d)$  and integers  $m(1)$  through  $m(k+n)$  such that the following properties hold:

- (i)  $p(1) < p(2) < \dots < p(d) < m(1) < \dots < m(k+n)$
- (ii) All of the numbers in (i) are pairwise relatively prime.
- (iii)  $\prod_{i=1}^k m(i) > p(d) \prod_{i=1}^{k-1} m(k+n+1-i)$
- (iv)  $\prod_{i=1}^k m(i) < \prod_{j=1}^d p(j) \prod_{i=1}^{k-d+1} m(k+i)$

Denote the product of the  $p(i)$ s by  $P$  and the product of the  $k$  smallest  $m(i)$ s by  $M$ . Let  $V$  be the collection of all integers  $y$  such that  $0 \leq y < M$ . Let  $W$  be  $\mathbb{Z}/P\mathbb{Z}$ . Let the revealing map  $\pi$  be the evaluation mod  $P$ . We choose the shadows in the following way. Let  $w$  be an element of  $\mathbb{Z}/P\mathbb{Z}$ . Choose a random number  $A$  such that  $0 \leq y = x + AP < M$ . The  $i$ th shadow  $H(i)$  is the set of all integers  $z$  between zero and  $M$  such that  $y \equiv z \pmod{m(i)}$ . The intersection of any  $r$  shadows  $H(i_1), \dots, H(i_r)$  is, by the Chinese Remainder

Theorem, the set of all  $z$  between zero and  $M$  such that  $y \equiv z$  modulo the product of  $m(k+i_1)$  through  $m(k+i_r)$ .

(5.1) Proposition: The Asmuth-Bloom scheme is a  $(d, k, n)$  ramp scheme with  $l = d - k$ . Knowledge is gained in a regular way with knowledge of each shadow after  $d - k$  shadows in the sense that

- a) Knowledge of  $k - d$  shadows and knowledge of  $y$  modulo  $d - s + 1$  of the  $p(i)$ s gives us knowledge of  $w$ .
- b) Knowledge of  $k - d + s$  shadows does not give us knowledge of  $y$  modulo any  $d - s$  of the  $P(i)$ s.

Proof: First we note that if the  $m(i)$ s are relatively close to the  $p(i)$ s (which is guaranteed by part (iii) of the definition) then

$$\log \frac{|V|}{|W|} = \frac{k}{d}.$$

Now suppose we know  $k$  shadows, that is, suppose we know  $y$  modulo  $k$  of the  $m(k+i)$ s. Denote their product by  $B$ . By the Chinese Remainder Theorem, we know  $y \bmod B$ . Since  $B \geq M$ , there is only one number  $z$  such that  $0 \leq z < M$  and  $y \equiv z \bmod B$ , namely  $y$ . Thus the intersection of  $k$  shadows is a single point in  $\pi^{-1}(w)$ , and so the Asmuth-Bloom scheme satisfies part a) of Definition (2.1).

Next suppose that we know no more than  $k - d$  shadows, that is, that we know  $y$  modulo no more than  $k - d$  of the  $m(k+i)$ s. Denote their product by  $B$ . By the Chinese Remainder Theorem, we know  $y$  modulo  $B$ . By (iii) and (i)  $M/B > P$ , and since  $P$  is relatively prime to all the  $m(i)$ s, this means that the set of all integers  $z$   $z \equiv y \bmod B$  and  $z < M$  covers all congruence classes modulo  $P$ . Thus the restriction of  $\pi$  to the intersection of no more than  $k - d$  shadows is surjective, and so the Asmuth-Bloom scheme satisfies part (b) of Definition (2.1).

Next, suppose we know  $k - d + s$  shadows, where  $1 \leq s < d$ . Let  $B$  denote the product of the corresponding  $m(k+i)$ s. Let  $C$  denote the product of any  $d - s$  of the  $p(i)$ s. By (i) and (iii)  $M/B > C$ . From this fact and the fact that  $C$  and  $B$  are relatively prime, we can conclude that the set of all integers  $z$  such that  $z \equiv y \bmod B$  and  $z < M$  covers all congruence classes mod  $C$ . Thus, if we let  $p$  denote the projection from  $\mathbb{Z}/P\mathbb{Z}$  to  $\mathbb{Z}/C\mathbb{Z}$ , the restriction of  $p \circ \pi$

to the intersection of the shadows is surjective, giving us part b) of the proposition.

Finally, suppose that we know  $y$  modulo  $k-d-s$  of the primes and  $d-s+1$  of the  $m(k+i)$ s. Denote the product of the primes by  $C$  and the product of the  $m(k+i)$ s by  $B$ . By (i) and (iv)  $M < BC$ . Thus knowledge of  $y$  modulo  $BC$  gives us  $y$ , giving us part a) of the proposition.

We note that conditions (iii) and (iv) of the definition of the Asmuth/Bloom ramp scheme can be changed by requiring that

$$\prod_{i=1}^k m(i) > p(d) \cdot \prod_{i=1}^{k-1+t} m(k+n+1-i)$$

and

$$\prod_{i=1}^k m(i) < \prod_{j=1}^d p(j) \cdot \prod_{i=1}^{k-d+t} m(k+i)$$

for some positive integer  $t < d$ . This would lessen the amount of information gained with each shadow. Knowledge of  $k-d+s$  shadows and  $d-s+t$  of the  $p(i)$ s would be required for knowledge of  $w$ . Or we could leave part (iii) of the definition unchanged: this would have the advantage of allowing us more leeway in choosing the  $m(i)$ s, and would mean that knowledge of  $k-d+s$  shadows and  $d-s+t$  of the  $p(i)$ s would always give us knowledge of  $w$ , while knowledge of fewer of the  $p(i)$ s sometimes would and sometimes would not, depending on the shadows. However, in both cases efficiency would be lost. The change in part (iii) of the definition would require the ratio  $m(1)/p(d)$  to be larger, while the change in part (iv) would allow the ratio to be larger.

(5.2) Theorem. Let  $R$  be a  $(d, k, n)$  Asmuth/Bloom ramp scheme. Let  $P$  denote the product of the  $p(i)$ s, let  $M$  denote the product of the  $k$  smallest  $m(i)$ s, and, for  $1 \leq r \leq d-k$ , let  $M_r$  denote the product of the  $r$  largest  $m(i)$ s. If  $r \leq k-d$  then  $R$  is Shannon  $t$ -relatively secure with knowledge of  $r$  shadows if and only if

$$([M/M_r P] - 1/t)([M/P] - 1/t) > (1+t)/t^2.$$

If  $r > k-d$  then  $R$  is  $t$ -relatively secure if and only if

$$[M/P] - 1/t > 0$$

Proof: By Lemma (2.6) it is enough to show that, if  $H$  is the intersection of  $r$  shadows, then

$$(5.3) \quad \frac{|\pi^{-1}(w)|}{(1+t) |\pi^{-1}(w^*)|} < \frac{|\pi^{-1}(w) \cap H|}{|\pi^{-1}(w^*) \cap H|} < \frac{(1+t) |\pi^{-1}(w)|}{|\pi^{-1}(w^*)|}$$

for all  $w$  and  $w^*$  in  $\pi(H)$ . Let  $H$  be the intersection of  $r$  shadows. Then  $H$  is the set of all  $z$  between zero and  $M$  such that  $z \equiv y \pmod{B}$ , where  $B$  is the product of the  $r$   $m(k+i)s$  corresponding to the  $l$  shadows. The cardinality of  $\pi^{-1}(w)$  for a given  $w$  in  $Z/PZ$  is either  $[M/P]$  or  $[M/P] + 1$ , and the cardinality of  $\pi^{-1}(w) \cap H$  is either  $[M/BP]$  or  $[M/BP] + 1$ , where  $[ ]$  denotes the greatest integer function. We consider the two cases:

A:  $r \leq k-d$  (in which case  $[M/BP] > 0$ ), and

B:  $r > k-d$  (in which case  $[M/BP] = 0$ ).

Case A. Suppose that  $l \leq k-d$ . Then  $[M/BP] > 0$ . The worst possible case as far as the right half of inequality (5.3) is concerned is

$$\begin{aligned} |\pi^{-1}(w)| &= [M/P], \\ |\pi^{-1}(w^*)| &= [M/P] + 1 \\ |\pi^{-1}(w) \cap H| &= [M/BP] + 1, \text{ and} \\ |\pi^{-1}(w^*) \cap H| &= [M/BP]. \end{aligned}$$

We are thus reduced to proving the inequality

$$([M/BP] + 1)/[M/BP] < (1+t)[M/P]/([M/P] + 1).$$

This is equivalent to

$$([M/BP] - 1/t)([M/P] - 1/t) > (1+t)/t^2.$$

Since  $B \leq N_r$  and can be equal to  $N_r$  it is thus necessary and sufficient to have

$$([M/N_r P] - 1/t)([M/P] - 1/t) > (1+t)/t^2.$$

The proof of the left-hand side of inequality (5.3) is similar.

Case B. Suppose that  $l > k-d$ . By conditions (i) and (iv) of the definition of the Asmuth/Bloom scheme we then have  $[M/BP] = 0$ . Thus the cardinality of  $\pi^{-1}(w) \cap H$  is either 0 or 1. Since we are only interested in proving the inequality for  $w$  and  $w^*$  in  $\pi(H)$ , the worst possible case as far as the right half of inequality (4.3) is

$$\begin{aligned} |\pi^{-1}(w)| &= [M/P], \\ |\pi^{-1}(w^*)| &= [M/P] + 1, \\ |\pi^{-1}(w) \cap H| &= 1, \end{aligned}$$

and

$$|\pi^{-1}(w^*) \cap H| = 1.$$

We thus have to prove the inequality

$$\frac{1}{[M/P]} < \frac{(1-t)}{[M/P] + 1}$$

This is equivalent to  $[M/P] - 1/t > 0$ .

There are two apparent contradictions here that need to be resolved. First we might ask the question: what if we choose  $t$  such that  $[M/P] - 1/t < 0$ ? Then would it be possible that

$$(4.4) \quad ([M/N_{rP}] - 1/t)([M/P] - 1/t) < (1+t)/t^2$$

thus giving Shannon  $t$ -relative security for small  $t$  but not possibly for larger  $t$ ? The answer is no. For suppose that  $[M/P] - 1/t < 0$  and that equation (4.4) holds. Then we have

$$\begin{aligned} ([M/N_{rP}] - 1/t)([M/P] - 1/t) &= t^2[M/N_{rP}][M/P] - t([M/N_{rP}] + [M/P] + 1) \\ &> (t+1)/t^2 \end{aligned}$$

Hence

$$t^2[M/N_{rP}][M/P] > t([M/N_{rP}] + [M/P] + 1)$$

and so

$$\frac{1}{t} < \frac{[M/N_r P][M/P]}{[M/N_r P] + [M/P] + 1}$$

But

$$[M/N_r P] - \frac{[M/N_r P][M/P]}{[M/N_r P] + [MP] + 1} = \frac{[M/N_r P]^2 + [M/N_r P]}{[M/N_r P] + [MP] + 1} > 0.$$

Thus  $[M/N_r P] - 1/t > 0$ , and so  $[M/P] - 1/t > 0$ .

The other apparent contradiction is that the requirements for  $t$ -relative security after knowledge of  $k$ - $d$  shadows is less stringent than the requirement before, making it seem that we lose information as we gain knowledge of the shadows. But this contradiction appears only if we forget the fact that after  $k$ - $d$  shadows some probabilities, which are not figured in the computations of  $t$ -relative security, go to zero. Thus, in spite of appearances, we still have more information than before.

Note that Shannon  $t$ -relative security and Condition (iii) of the definition of the Asmuth/Bloom scheme require that  $M$  be large in comparison to  $PN_r$ , while Condition (iv) requires that  $M$  be relatively small. If  $d$  is large then Shannon  $t$ -relative security and Condition (iii) become relatively easy to obtain, while Condition (iv) becomes harder. The reverse is true if  $d$  is small.

The difficulty seems to lie in the inequality in Condition (iii). If this inequality could be replaced by an equality, then it and the first two conditions would suffice to give us a ramp scheme. Condition (iv). This indeed can be done in certain cases of the generalized Asmuth/Bloom scheme, in which the integers are replaced by a Euclidean domain and the  $p(i)$ s and  $m(i)$ s can be replaced by relatively prime elements of the same degree. However, since the only known practical example of such a scheme is Shamir's scheme [SH79], which was discussed in Section 3, we refrain from discussing generalized Asmuth-Bloom schemes here.

This work was supported in part by NSA Grant MDA-83-H-0002.

#### REFERENCES

- AS81 C. A. Asmuth and G. R. Blakley, An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems, Computers and Mathematics with Applications, Vol. 7 (1981), pp. 447-450.



- AS82 C. A. Asmuth and G. R. Blakley, Pooling, splitting and restituting information to overcome total failure of some channels of communication, Proceedings of the 1982 Symposium on Security and Privacy, IEEE Computer Society, Los Angeles, (1982), pp. 156-169.
- AS83 C. Asmuth and J. Bloom, A modular approach to key safeguarding. IEEE Transactions on Information Theory, Vol. IT-30 (1983), pp. 208-210.
- BL79 G. R. Blakley, Safeguarding cryptographic keys, Proceedings of the National Computer Conference, 1979, American Federation of Information Processing Societies -- Conference Proceedings, Vol. 48 (1979), AFIPS Press, Montvale, New Jersey (1979), pp. 313-317.
- BL80 G. R. Blakley, One-time pads are key safeguarding schemes, not cryptosystems. Fast key safeguarding schemes (threshold schemes) exist, Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society, New York (1980), pp. 108-113.
- BL81a G. R. Blakley and Laif Swanson, Security proofs for information protection systems, Proceedings of the 1981 Symposium on Security and Privacy, IEEE, Computer Society, New York (1981), pp. 75-88.
- BL81b J. Bloom, A note on superfast threshold schemes, Preprint, Texas A&M University, Department of Mathematics (1981)  
and  
Threshold schemes and error correcting codes, Abstracts of Papers Presented to the American Mathematical Society, Vol. 2 (1981), p. 230.
- BL82 G. R. Blakley, Protecting information against both destruction and unauthorized disclosure, Proceedings of the 1982 Carnahan Conference on Security Technology, University of Kentucky Press, (1982), pp. 123-133.
- BL83 G. R. Blakley and L. Swanson, Infinite structures in information theory, in D. Chaum, R. L. Rivest and A. T. Sherman, Advances in Cryptology, Proceeding of Crypto '82, Plenum Press, New York (1983), pp. 39-50.
- CH79 D. Chaum, Computer systems established, maintained, and trusted by mutually suspicious groups, Memorandum No. UCB/ERL/M79/10, UC Berkeley ERL (1979).
- CH82 D. Chaum, Computer systems established, maintained and trusted by mutually suspicious groups, Ph.D. dissertation in Computer Science, UC Berkeley (1982).

- DA80 G. I. Davida, R. A. DeMillo and R. J. Lipton, Protecting shared cryptographic keys, Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society, New York (1980), pp. 100-102.
- DE82 D. E. R. Denning, Cryptography and Data Security, Addison-Wesley, Reading, Massachusetts (1980).
- HA83 S. Harari, Secret sharing systems, in Secure Digital Communications, Edited by G. Longo, Springer-Verlag, Wien (1983), pp. 105-110.
- HO71 K. Hoffman and R. Kunze, Linear Algebra, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey (1971).
- KA83 E. D. Karnin, J. W. Greene and M. E. Hellman, On secret sharing systems, Verbal presentation, Session B3 (Cryptography), 1981 IEEE International Symposium on Information Theory, Santa Monica, California, February 9-12 (1981),  
and  
On secret sharing systems, IEEE Transactions on Information Theory. Vol. IT-29 (1983), pp. 35-41.
- KO81 A. G. Konheim, Cryptography: A Primer, Wiley-Interscience, New York (1981).
- KO85 S. Kothari, On a Generalized Threshold Scheme, Proceedings of Crypto '84, Springer-Verlag, New York (1985).
- LI83 R. Lidl and H. Niederreiter, Finite Fields, Vol. 20 of the Encyclopedia of Mathematics and its Applications, Addison-Wesley, Reading, Massachusetts (1983).
- MA67 S. MacLane and G. Birkhoff, Algebra, MacMillan, New York, (1967).
- MA78 F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes, North-Holland, Amsterdam (1978).
- MC81 R. J. McEliece and D. V. Sarwate, On sharing secrets and Reed-Solomon codes, communications of the ACM, Vol. 24 (1981), pp. 583-584.
- MI87 M. Mignotte, How to share a secret, in Cryptography, Edited by T. Beth, Springer-Verlag, Berlin (1983), pp. 371-375.
- OZ84 L. H. Ozarow and A. D. Wyner, Wire-tap channel, II, AT&T Bell Labs Technical Journal, vol. 63 (1984), to appear.
- SH79 A. Shamir, How to share a secret, Communications of the ACM, Vol. 22 (1979), pp. 612-613.

Selim G. Akl & Henk Meijer

A Fast Pseudo Random Permutation Generator  
With Applications to Cryptology

Department of Computing and Information Science  
Queen's University  
Kingston - Ontario - Canada

## 1. Introduction

Pseudo random sequences of integers are most commonly generated by linear congruence methods [5] or by linear shift registers [1]. These sequences can be used in cryptology if they are cryptographically secure [9]:

A pseudo-random sequence is cryptographically secure if given any segment of the sequence, it is computationally infeasible to compute other segments of the sequence.

It has been shown that sequences generated by linear congruence methods or by linear shift registers are not cryptographically secure [1,6,7]. By using one-way functions [2,3,9,10,11] or cryptographically strong encryption algorithms like the DES secure pseudo random number sequences can be constructed.

In this paper we propose a new class of generators. Members of the new class compare favourably with existing cryptographically strong generators since

- they are fast, even in software implementation, and
- they can quickly and easily be programmed in any computer language.

---

This work was supported by the Natural Sciences and Engineering Research Council of Canada under Strategic Grant G0381.

We first explain how the generators can be constructed in general. In sections 3 and 4 the generation of pseudo random permutations is described in detail. In section 5 it is shown how these permutations may be used in cryptology.

## 2. General scheme

Let  $S$  be a set with cardinality  $\#S$  and let  $*$  and  $\circ$  be two binary operations on  $S$ . We construct a sequence  $\{s^i\} = s^0, s^1, s^2, \dots$  as follows:

Step 0 : Let  $x^0$  be an arbitrary element of  $S$ .

Step 1 :  $x^i = x^{i-1} * q$ , for  $i > 0$  and some  $q \in S$ .

Step 2 :  $s^i = x^{ki} \circ x^{ki+1} \circ \dots \circ x^{ki+k-1}$ ,  
for all  $i \geq 0$  and some positive integer  $k$ . []

The element  $q$  should be chosen such that the sequence  $\{x^i\}$  has a large period. In the best case  $\{x^i\}$  will have a period equal to  $\#S$ , which implies that the period of the sequence  $\{s^i\}$  is

$$\frac{\#S}{\gcd(k, \#S)}.$$

Obviously the set  $S$ , element  $q$ , integer  $k$  and operations  $*$  and  $\circ$  have to be chosen such that  $\{s^i\}$  becomes cryptographically secure. We will use the above algorithm to construct a sequence of pseudo-random permutations. Although we have not been able to prove our random permutation generator secure, we conjecture that the sequence is secure since in our generator

- i-  $q$  is chosen such that the period of  $\{x^i\}$  is maximal
- ii- the operations  $\circ$  and  $*$  do not have the distributive property
- iii- the set  $S$  with operation  $\circ$  forms a group.

Fact -i- ensures that the sequence does not cycle in practice if  $S$  is chosen such that  $\#S > 2^{200}$ ,

say. Fact -ii- makes it hard to simplify and solve equations that express elements of the sequence  $\{s^i\}$  in terms of the unknown quantities  $\{x^i\}$  and  $q$ . Finally fact -iii- implies that

$$a = b \circ x$$

has a solution  $x \in S$  for all  $a, b \in S$ . So if an element  $s^j$  of the sequence  $\{s^i\}$  is known, then little is known about the values of

$$x^{kj}, x^{kj+1}, \dots, x^{kj+k-1}.$$

### 3. Random permutations

Let  $P$  denote the set of all permutations of  $(0 \ 1 \ 2 \ \dots \ n-1)$ . Various algorithms that map the set  $Z_{n!}$  onto the set  $P$  exist [4,5]. In this paper we use the mapping given in [5] and we denote it by  $f$ . So

$$f : Z_{n!} \rightarrow P$$

is a bijection. We compute the pseudo random sequence of permutations  $\{p^i\}$  by the following algorithm:

- Step 0: Let  $x^0$  be an arbitrary element of  $Z_{n!}$  and  $q \in Z_{n!}$  with  $\gcd(n!, q) = 1$ .
- Step 1:  $x^i = (x^{i-1} + q) \bmod n!$ , for  $i > 0$ .
- Step 2:  $p^i = f(x^{ki}) \circ f(x^{ki+1}) \circ \dots \circ f(x^{ki+k-1})$  for all  $k$ ,  $i \geq 0$  and some positive integer  $k$ , where  $\circ$  is composition of permutations. []

Notice that the sequence generated in step 1 is a special case of the linear congruential sequence

$$x^i = a x^{i-1} + q \bmod m$$

with  $a=1$  and  $m=n!$ . By choosing  $q$  relatively prime to  $n!$ , we are guaranteed a sequence  $\{x^i\}$  of maximum period  $n!$  [5].

We will see in the next section how the above algorithm can be implemented efficiently.

#### 4. Implementation

Elements in  $Z_{n!}$  can be represented in a mixed-radix notation [5] as follows: if  $x \in Z_{n!}$  it can be written as

$(x_0 \ x_1 \ \dots \ x_{n-1})$  with

$$x = \sum_{i=0}^{n-1} x_i \cdot i!$$

and  $0 \leq x_i \leq i$ .

Addition modulo  $n!$  of two elements of  $Z_{n!}$ , written in this mixed-radix notation can be achieved with at most  $2n$  additions,  $n$  subtractions and  $n$  comparisons.

If we denote a permutation  $p \in P$  by the sequence  $(p(0) \ p(1) \ \dots \ p(n-1))$  then the mapping  $f$  from  $Z_{n!}$  to  $P$  can be defined by [5]:

function  $f(x)$ :

let  $p(i) = i$ , for  $i=0, \dots, n-1$

for  $i = n-1, n-2, \dots, 1, 0$  do

exchange  $(p(i), p(x_i))$

end for

return  $p$ . []

Since composition of permutations of size  $n$  can be achieved with  $n$  exchanges, it is obvious that each element of the sequence  $\{p^i\}$  can be computed in  $O(kn)$  steps.

#### 5. Generating bits

A pseudo random sequence of bits can be added bitwise modulo 2 to the bits of a message to construct a cryptogram. If the bit sequence is cryptographically secure, then so is

the resulting cryptogram. We show below how the random sequence of permutations can be used to produce a pseudo-random sequence of bits.

Let  $b^i$  be an arbitrary bit vector of length  $n$  and let  $p[b^i]$  denote the bit vector that is the result of applying permutation  $p$  to vector  $b^i$ . If  $\{b^i\}$  is an arbitrary pseudo-random sequence of bit vectors then we conjecture that  $\{p^i [b^i]\}$  is a cryptographically secure bit stream.

In the algorithm below we use sequence  $\{b^i\}$  defined by  
 $b^0 = (01010 \dots 0101)$   
 $b^i = p^{i-1}[b^{i-1}]$  ,  $i > 0$ .

We propose to use  $k=2$  and  $n=64$ . So the period of  $\{x^i\}$  and therefore of  $\{p^i [b^i]\}$  is approximately  $2^{300}$ .

The algorithm applies the permutations  $p^i = f(x^{2i}) \circ f(x^{2i+1})$  to the bit vectors  $b^i$  without in fact computing  $p^i$ , but rather by applying  $x^{2i}$  and  $x^{2i+1}$  directly to  $b^i$ . Implemented in the programming language C and running on a VAX 11/780, the algorithm generated more than 10 000 bits/second.

Algorithm to generate a bit stream:

```

n <- 64
k <- 2
x0, x1, ..., xn-1 <- initial value
q0, q1, ..., qn-1 <- integer relatively prime to n!
b0, b1, ..., bn-1 <- 0,1,0,1,0,1,...,0,1

repeat as many times as required
  do k times
    {compute x + q mod n!}
    carry <- 0
    for i = 1,2,..., n-1 do
      xi <- xi + qi + carry
      if xi > i then xi <- xi - (i+1)
      carry <- 1

```

```

else carry <- 0
end for
{apply x to b }
for i = n-1, n-2,..., 1 do
    exchange ( $b_i$ ,  $b_{x_i}$ )
end for
end do
output  $b_0, b_1, \dots, b_{n-1}$ 
end repeat. []

```

Notice that the greatest common divisor of two numbers written in mixed radix-notation can easily be computed using the Euclidean algorithm.

## 6. Conclusion

We applied the statistical tests for randomness described in [1] to the bitstream generated by the algorithm described above. For these tests we chose  $n=64$ ,  $k=2$  and random values for  $q$  and we used an input bitstream of vectors  $\{b^i\}$  given by

$$b^i = 111\dots 1000\dots 0 \quad \text{for } i=0,1,2,\dots$$

where the number of 1's in each vector  $b^i$  has a binomial distribution with parameters 64 and  $1/2$ . The generated bitstreams did not exhibit any statistical weaknesses.



## References

- [1] H. Beker and F. Piper, Cipher Systems, John Wiley, 1982.
- [2] L. Blum, M. Blum and M. Shub, Comparison of two pseudo-random generators, Proceedings of Crypto 1982.
- [3] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo random bits, FOCS, 1982.
- [4] G.D. Knott, A numbering system for permutations and combinations, Communications of the ACM Vol. 19, No. 6, June 1976.
- [5] D.E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms, Addison-Wesley, 1981.
- [6] D.E. Knuth, Deciphering a linear congruential encryption, Technical Report 024800, Stanford University, 1980.
- [7] J.B. Plumstead, Inferring a sequence generated by a linear congruence, FOCS, 1982.
- [8] J. Reeds, Cracking a random number generator, Cryptologia, Vol. 1, January 1977.
- [9] A. Shamir, On the generation of cryptographically strong pseudorandom sequences, ACM Transactions on Computer Systems, Vol. 1, No. 1, February 1983.
- [10] U.V. Vazirani and V.V. Vazirani, Trapdoor pseudo-random number generators with applications to protocol design, FOCS, 1983.
- [11] A. Yao, Theory and applications of trapdoor functions, FOCS, 1982.

# ON THE CRYPTOGRAPHIC APPLICATIONS OF RANDOM FUNCTIONS

(EXTENDED ABSTRACT)

Oded Goldreich, Shafi Goldwasser, Silvio Micali

Laboratory for Computer Science

M.I.T.

Cambridge, MA 02139

## ABSTRACT

Now that "random functions" can be efficiently constructed ([GGM]), we discuss some of their possible applications to cryptography:

- 1) Distributing unforgeable ID numbers which can be locally verified by stations which contain only a small amount of storage.
- 2) Dynamic Hashing: even if the adversary can change the key-distribution depending on the values the hashing function has assigned to the previous keys, still he can not force collisions.
- 3) Constructing deterministic, memoryless authentication schemes which are provably secure against chosen message attack.
- 4) Construction Identity Friend or Foe systems.

The first author was supported in part by a Weizmann Postdoctoral fellowship. The second author was supported in part by the International Business Machines Corporation under the IBM/MIT Joint Research Program, Faculty Development Award agreement dated August 9, 1983.

## 1. INTRODUCTION

In our paper "How to Construct Random Functions" ([GGM]), we have

- 1) Introduced an algorithmic measure of the randomness of a function. (Loosely speaking, a function is random if any polynomial time algorithm, which asks for the values of the function at various points, cannot distinguish the case in which it receives the true values of the function, from the case in which it receives the outcome of independent coin flips.)
- 2) Constructed functions that are easy to evaluate and, nevertheless, achieve maximum algorithmic randomness, under the assumption that there exist one-way permutations.

In this paper, we describe in details 4 cryptographic applications of these "pseudo-random functions": Storageless ID Number Distribution, Dynamic Hashing, Deterministic Private-key Signature Scheme and Identify Friend or Foe. Before describing these applications, let us recall some of the definitions which appeared in [GGM].

### 1.1 Poly-Random Collections

Let  $I_k$  denote the set of all  $k$ -bit strings. Consider the set,  $H_k$ , of all functions from  $I_k$  into  $I_k$ . Note that the cardinality of  $H_k$  is  $2^{k^2}$ . Thus to specify a function in  $H_k$  we would need  $k^2$  bits: an impractical task even for a moderately large  $k$ . Even more, assume that one randomly selects subsets  $H_k^\# \subseteq H_k$  of cardinality  $2^k$  so that each function in  $H_k^\#$  has a unique  $k$ -bit index; then there is no polynomial time Turing Machine that, given  $k$ , the index of a function  $f \in H_k^\#$  and  $x \in I_k$ , will evaluate  $f(x)$ .

Our goal is to make "random functions" accessible for applications. I.e. to construct functions that can be easily specified and evaluated and yet cannot be distinguished from functions chosen at random in  $H_k$ . Thus we restrict ourselves to choose functions from a subset  $F_k \subseteq H_k$  where the collection  $F = \{F_k\}$  has the following properties:

- 1) Indexing: Each function in  $F_k$  has a unique  $k$ -bit index associated with it. (Thus picking randomly a function  $f \in F_k$  is easy.)
- 2) Poly-time Evaluation: There exists a polynomial time Turing machine that given an index of a function  $f \in F_k$  and an input  $x$ , computes  $f(x)$ .
- 3) Pseudo-Randomness: No probabilistic algorithm that runs in time polynomial in  $k$  can distinguish the functions in  $F_k$  from the functions in  $H_k$ .

More precisely: if the collection  $F$  passes all polynomial time statistical tests for functions, where the notions "statistical test for functions" and "passes a test" are hereby defined.

A polynomial time statistical test for function is a probabilistic polynomial time algorithm  $T$  that, given an input  $k$  and access to an oracle  $O_f$  for a function  $f: I_k \rightarrow I_k$ , outputs either 0 or 1. Algorithm  $T$  can query the oracle  $O_f$  only by writing on a special query-tape some  $y \in I_k$  and will read the oracle answer,  $f(y)$ , on a separate answer-tape. As usual,  $O_f$  prints its answer in one step.

Let  $F = \{F_k\}$  be a collection of functions. We say that  $F$  passes the test  $T$  if for any polynomial  $Q$ , for all sufficiently large  $k$ :

$$\left| p_k^F - p_k^R \right| < \frac{1}{Q(k)}$$

where  $p_k^F$  denotes the probability that  $T$  outputs 1 on input  $k$  and access to an oracle for a function  $f$  randomly chosen in  $F_k$ .  $p_k^R$  is the probability that  $T$  outputs 1 when given the input parameter  $k$  and access to an oracle  $O_f$  for a function  $f$  randomly picked in  $H_k$  (i.e. a random function).

Such a collection of functions  $F$  will be called a poly-random collection. Loosely speaking, despite the fact that the functions in  $F$  are easy to select and easy to evaluate, they will exhibit, to an examiner with polynomially bounded resources, all the properties of randomly selected functions.

The above definition is highly constructive. In [GGM] it was shown how to transform any one-to-one one-way function to an algorithm  $A_f$  for a poly-random collection of functions  $F$ . The construction is in two steps: first, using Yao's construction (see Appendix A) to transform a one-to-one one-way function into a Cryptographically Strong Pseudo Random Bit generator (CSPRB-generator); next, using ANY CSPRB-generator to construct a poly-random collection (see Appendix B). However, for practical purposes we will consider only poly-random collections whose underlying CSPRB generator is fast.

Efficiency considerations

In the recent years many CSPRB generators have been proposed ([BBS], [BM], [GMT], [Y]), based on various intractability assumptions and demonstrating various degrees of practicality.

Using the new results of Chor and Goldreich [CG] it is now possible to construct fast CSPRB generators which are "equivalent" to factoring: On input a  $k$ -bit long seed, these generators output  $\log k$  bits at the price of one modular multiplication of two  $k$ -bit long integers. Factoring  $k$ -bit long integers is  $\text{poly}(k)$  reducible to distinguishing the sequence generated by these generators from truly random sequences.

Let  $T$  denote the average time needed for generating one bit in the

underlying generator used in our construction of a poly-random collection. Then, evaluating a function chosen at random from  $F_k$  can be done in time  $O(k^2T)$ .

## 1.2 Comparison with CSPRB generators

The fundamental definitions and properties of Cryptographically Strong Pseudo-Random Bit(CSPRB) generators are given in Appendix A.

It is a theoretical challenge and an extremely useful task to find the most general properties of randomness that can be achieved by efficient pseudo-random programs.

Let us consider the effect of such programs on probabilistic computation.

CSPRB generators cut down the number of coin tosses

Performing a probabilistic polynomial-time computation that requires  $k^t$  random bits is trivial if we are willing to flip  $k^t$  coins. A very interesting feature of CSPRB generators is that they guarantee the same result of the computation by flipping only  $k$  coins!

Poly Random Collection cut down the storage as well

The existence of poly-random collections allows to successfully replace a random oracle(function), in any polynomial time computation, by  $k$  random bits.

It should be noticed that computing with a random oracle is different from computing with a coin. In fact, the bit associated with each string  $x$ , not only is random, but does not change in time and is stored for free! The advantages of the random oracle model are clarified by all the applications discussed in the following sections.

Again, it is trivial(see Appendix C) to simulate a computation with a random oracle (function) that is queried on  $k^t$  inputs if one is willing to use  $k^t$  bits of storage. A very interesting feature of poly-random collections is that they guarantee the same result of the computation by using only  $k$  bits of storage!

Sharing Randomness in a distributed environment

An additional advantage of our solution is that it enables many parties to efficiently share such an  $f$  in a distributed environment. By sharing  $f$  we mean that if  $f$  is evaluated at different times by different parties on the same input  $x$ , the same value  $f(x)$  will be obtained. Such sharing is efficient as it can be achieved by an initial step which consists of (1) One party flips  $k$  coins; and (2) All parties record the result. After this initial step no more coin flips or message exchanges are needed. The  $k$  bits stored by all determine a shared function of the poly-random collection.

Assume that in "situation"  $S$ , some party (processor)  $p_j$  wants to make a random choice so that the other processors will know it. Then it will simply compute  $f(j, S)$ . Because of the "randomness" of  $f$ , such choices are as good as truly random choices. Note that any other processor  $p_i$  can compute the random choices processor  $p_j$  did in situation  $S$ , without any extra communication!

## 2. "STORAGELESS" DISTRIBUTION OF SECRET NUMBERS

### 2.1 The Problem

Consider the problem in distributing secret identification numbers (ID's). Every user in the system should receive a secret ID from the system, which is easily verifiable by the system, but hard to compute by anyone else. An example may be assigning calling card numbers to telephone customers. We assume there are no two users with the same name.

A possible solution could be to assign each user  $U$  a secret randomly selected number  $r$ , and store the pair  $(U, r)$  in a protected data base. This solution requires storage proportional to the number of users, which may be very large. Using our random functions, we propose a "storageless" solution to this problem.

### 2.2 Our Solution

Let  $F_k$  be a poly-random collection, and let the server pick  $f \in F_k$  at random. Then, the server assigns each user  $U$ ,  $f(U)$  as her secret number. To verify whether  $(U, n)$  is a legal pair, the server computes  $f(U)$  and compares it with  $n$ . Now, suppose that Alice has such a secret ID and that all of her relatives  $(A_1, A_2 \text{ etc.})$ , who possess their own secret ID's gang up to discover Alice's ID. They try to exploit the fact that their names  $A_1, A_2, \dots, A_q$  are similar to hers. However, for  $f$  picked by the server from a poly-random collection, they could not compute  $f(\text{Alice})$  given  $f(A_1), \dots, f(A_q)$ .

This solution requires only  $k$  bits of storage, when the number of users in the system is bounded by a polynomial in  $k$ .

Notice that this solution also works in a distributed environment. If each "branch" of the server has a computer with the (shared  $k$ -bit) secret  $s$  embedded in it, a secret number can be handed out in San Francisco and be (locally) verified in Boston.

### 2.3 The Correctness Argument: Simulation

Assume that one-way permutations exists and that  $g$  is such a permutation. Let  $F = \{F_k\}$  be a poly-random collection constructed using  $g$  and let  $f$  be a function randomly selected from  $F_k$ .

Assume that  $A_1, A_2, \dots, A_q$  have some advantage in guessing  $f(\text{Alice})$  from  $f(A_1), \dots, f(A_q)$ . Clearly, they could not have such an advantage if  $f$  were a truly random function. Thus, they can distinguish  $f$  from a truly random function. This, in turn, provides an algorithm for inverting  $g$ .

### 3. DYNAMIC HASHING

#### 3.1 The problem

Consider the problem of hashing a few long keys (names) into shorter addresses (abbreviations), such that with very small probability two keys are hashed into the same address.

The classical purposes of hashing are:

- (1) To save on memory space. (For example, assigning physical memory location to variables can be done by applying a hashing function to the variable names. This way there is no need to store the variable names, which may be long.)
- (2) To allow fast retrieval of keyed information (hashing will help in applications where accessing the memory is slower than evaluating the function).

#### 3.2 Our solution

In order to present our solution let us first generalize the definition of a poly-random collection. Let  $p_1(\cdot)$  and  $p_2(\cdot)$  be two polynomials. A generalized poly-random collection is a collection,  $F = \{F_{p_1}(k), p_2(k)\}$ , of indexed and easy to evaluate functions from  $I_{p_1}(k)$  into  $I_{p_2}(k)$  such that a function chosen at random from  $F_{p_1}(k), p_2(k)$  cannot be distinguished in  $\text{poly}(k)$  time from a random function from  $I_{p_1}(k)$  into  $I_{p_2}(k)$ .

Our solution consists of using a function  $f$  chosen at random from  $F_{p_1}(k), p_2(k)$  as a hashing function. (i.e. key  $K$  is hashed into address  $f(K)$ ).

Note that our hashing function is much more robust with respect to polynomial time computation than the Universal Hashing suggested by Carter and Wegman [CW]. In their scheme, the adversary picks an arbitrary key distribution and the hashing performance (expected number of collisions) is analyzed with respect to this fixed distribution.

Our scheme performs well even if the adversary does not fix his key distribution a priori, but can dynamically change the key distribution during the hashing process upon seeing the hashing function values on previous keys. In other words, even if an adversary can pick the keys to be hashed and examine the values of the hash func-

tions on old keys, he cannot force collisions. Moreover, the adversary cannot distinguish the hashing value for a new key from a random value.

The robustness of our hashing technique, makes it particularly suitable for cryptographic purposes. For example, Brassard ([B]) has pointed out the advantages of authentication schemes based on "cryptographically strong" hashing functions. This theme is further developed in section 5.

#### 4. MESSAGE AUTHENTICATION AND TIME-STAMPING

In this section we will show how to construct deterministic, memoryless, authentication schemes which are highly robust, as discussed in the following concrete setting.

Assume that all the employees of a large bank communicate through a public network. As an adversary may be able to inject messages, the employees need to authenticate the messages they send to each other (e.g. "transfer sum  $S$  from account  $A$  to account  $B$ "). A solution may consist of appending to the message  $m$  an authentication tag which is hard to compute by an adversary. In particular, we propose the following. Let all employees have access to authentication machines which compute a function  $f_S$  in a poly-random collection. The tag associated with a message  $m$  is  $f_S(m)$ . We can tradeoff security for the length of the tag. For example, if one uses only the first 20 bits of  $f_S(m)$  as an authentication tag, then the chance that an adversary could successfully authenticate a message is about 1 in a million.

To avoid playback of previously authenticated messages, it is common practice to use time-stamps. Namely, authenticate  $m$  concatenated with date it was sent. So far, time-stamping was only a heuristic as an adversary who sees the message  $m$  authenticated with date  $D$  could conceivably authenticate  $m$  with another date (say  $D+1$ ). Using our solution for message authentication, time-stamping makes playback provably hard. This is the case as for a random function  $f(x)$  is totally unrelated to  $f(x+1)$ , and therefore the same holds (with respect to polynomial-time adversaries) for poly-random collections.

Another threat to the Bank's security is the loyalty of its own employees. They have the authenticating computer at their disposal and can use it to launch a chosen message attack against the scheme, so that when they are fired they can forge transactions. Our message authentication scheme remains secure even when the employees are not trustworthy, if each message to be authenticated is automatically



time stamped by the computer. An employee who leaves the bank, after having widely experimented with the machine, will not be able to authenticate even one new message.

##### 5. AN IDENTIFY FRIEND OR FOE SYSTEM

The members of a large but exclusive society are well known for their brotherhood spirit. Upon meeting each other, anywhere in the world, they extend hospitality, favors, advice, money, etc. Naturally they face the danger of imposters trying to take advantage of their generosity. Thus, upon meeting each other, they must execute a protocol for establishing membership. As they meet in public places (busses, trains, theatre), they must be careful not to yield information that can lead to future successful impersonations. They go around carrying pocket computers on which they may make calculations.

Clearly a password scheme will not suffice in this context, as the conversations are public. An interactive identification scheme is needed where the ability to ask questions does not enable future successful impersonations. Note that the questions that A may ask member B, must be picked from an exponential range to prevent an active imposter from asking all possible questions, receiving all possible answers and thereafter successfully impersonating as a member (or to prevent a passive imposter from having a non-negligible probability of being asked a question that he overheard the answer to).

Using our poly-random collection, we can fully solve this problem. Let the president of the society choose a  $k$ -bit random string  $s$ , specifying a function  $f_s$  in a poly-random collection. Each member receives a computer which calculates  $f_s$ . When member A meets B, he asks " $z$ ?" where  $z \in R_{I_K}$ . Only if B answers  $f_s(z)$ , will member A be convinced that B is a member. In addition, if the computers that calculate  $f_s$  can be manufactured so that they can not be duplicated, then losing a computer does not compromise the security of the entire scheme; it just allows one non-member to enjoy the privileges of the society.

##### 6. SOLVING BLUM BLUM & SHUB OPEN PROBLEM

Blum, Blum and Shub [BBS] have presented an interesting CSPRB generator whose sequences pass all polynomial time statistical tests if and only if deciding Quadratic Residuosity modulo a Blum-integer<sup>(1)</sup> whose factorization is not known, is intractable.

<sup>(1)</sup> A Blum integer is an integer of the form  $p_1 p_2$  where  $p_1$  and  $p_2$  are distinct primes both congruent to 3 mod 4.

Notice that, even though a CSPRB sequence generated with a  $k$ -bit long seed is  $P_1(k)$ -bit long, a CSPRB generator and a seed  $s$  define an infinite bit-sequence  $b_0, b_1, \dots$ . An interesting feature of Blum Blum Shub's generator is that knowledge of the seed and of the factorization of the modulus allows direct access to each bit in an exponentially long bit string (i.e. if  $k$  denotes the length of the seed and  $i \leq k$ , then the  $i$ -th bit in the string ( $b_i$ ) can be computed in  $\text{poly}(k)$  time). This is due to the special weak one-to-one one-way function on which the security of their generator is based. However, this exponentially long bit string MAY NOT appear "random". Blum, Blum and Shub only prove that any SINGLE polynomially long interval of CONSECUTIVE bits in the string passes all polynomial time statistical tests for strings. Indeed, it may be the case that, given  $b_1, \dots, b_k$  and  $b_{2^{\sqrt{k}}+1}, \dots, b_{2^{\sqrt{k}}+k}$  it is easy to compute any other bit in the string. Another CSPRB generator which possess the direct access property was suggested by Goldwasser, Micali and Tong GMT. Their generator is also based on a specific intractability assumption (factoring in a subset (of half) of the Blum integer). Also, it is not known whether direct access in the GMT generator preserves randomness.

The Blum Blum Shub open problem consists of whether direct access to exponentially far away bits in their pseudo-random pad is a "randomness preserving" operation. Or more generally, whether there exist generators which possess such a "randomness preserving direct access" property.

The Blum Blum Shub's generator, when fed with a  $k$ -bit long seed  $s$ , defines a function  $f_s$  in the following way: for each  $k$ -bit integer  $x$ ,  $f_s(x)$  is the  $x$ -th block of  $k$  bits in the pad. I.e.  $f_s(x) = b_{k \cdot x + 1} \dots b_{k \cdot x + k}$ . Recall that the Blum Blum Shub generator is based on the intractability assumption of a special permutation and furthermore, even under this assumption, direct access was not proved to be a randomness preserving operation. As a consequence  $f_s$  may not be "random".

We solve the above problem in a very strong sense. In fact we construct random functions  $f$ , from  $k$ -bit strings into  $k$ -bit strings, given ANY one-way permutation. Having constructed such an  $f$ , we have virtually constructed the  $k2^k$ -bit long string  $s_f = f(1)f(2)\dots f(2^k)$ . For the set  $\{s_f\}$  we prove that direct access is a "randomness preserving" property.

## APPENDICES

## Appendix A: CSPRB Generators, One-Way Permutations and Yao's Construction.

Following the unpredictable number generators of Shamir [S], Blum and Micali [BM] have introduced the notion of Cryptographically Strong Pseudo-Random Bit (CSPRB) generators. They have also presented the first instance of it, relying on the intractability assumption of the discrete logarithm problem.

Let  $t$  be any fixed constant. A CSPRB generator is a deterministic program that receives as input a (random)  $k$ -bit long seed and outputs a  $k$ -bit long (pseudo-random) sequence such that the next bit in the sequence cannot be predicted in polynomial (in  $k$ ) time from the preceding bits. Yao [Y] introduces the notion of a polynomial-time statistical test and shows that the outputs of CSPRB generators pass all polynomial-time statistical tests. He also proves that one can construct CSPRB generators given any (weak) one-way permutation.

Let us be more formal. Let  $f_k: I_k \rightarrow I_k$  be a sequence of permutations such that there is a polynomial-time algorithm that on input  $x \in I_k$  computes  $f_k(x)$ . Let the function  $f$  be defined as follows:  $f(x) = f_k(x)$  if  $x \in I_k$ . We say that  $f$  is a one-to-one one-way function if for all polynomial-time Turing Machines  $M$  there is a polynomial  $P$  such that, for all sufficiently large  $k$

$$M(x) \neq f_k^{-1}(x) \text{ for at least a fraction } \frac{1}{P(k)} \text{ of the } x \in I_k.$$

LEMMA 1(Yao [Y]): Given a weak one-to-one one-way function, it is possible to construct CSPRB generators.

Sketch of the proof: Given a one-way permutation,  $f$ , Yao constructs a hard to evaluate predicate by taking the exclusive-or of the inverse of  $f$  on polynomially many points. Namely,

$$B_k(x_1, x_2, \dots, x_{kt}) = \text{XOR } f_k^{-1}(x_1) f_k^{-1}(x_2) \dots f_k^{-1}(x_{kt})$$

where XOR  $s$  is the exclusive-or of all the bits of the string  $s$ .

Appendix B: The Construction of  $F$  (from any CSPRB Generator) (GGM)

Let  $G$  be a CSPRB-generator. Recall that  $G$  is a function defined on all bit strings such that if  $x \in I_k$ ,  $G(x) = b_1^x, \dots, b_{P(k)}^x$ . With no loss of generality, we can assume that  $P(k) \geq 2k$ .

(This is the case since Goldreich and Micali (GM) have shown that the existence of a CSPRB generator which expands a  $k$ -bit long seed into a  $(k+1)$ -bit output pad, yields the existence of a generator which expands a  $k$ -bit long seed into a  $2k$ -bit long pad).

Let  $S = \bigcup_k S_k$  be defined as follows.  $S_k$  is the set of all the first

2k bits output by G on seeds of length k. Then S passes all polynomial time statistical tests for strings.

Let  $x \in I_k$  be a seed for G.  $G_0(X)$  denotes the first k bits output by G on input X;  $G_1(X)$  denotes the next k bits output by G. Let  $\alpha = \alpha_1 \alpha_2 \dots \alpha_t$  be a binary string. We define  $G_{\alpha_1 \alpha_2 \dots \alpha_t}(x) = G_{\alpha_t}(\dots(G_{\alpha_2}(G_{\alpha_1}(x)))\dots)$ .

Let  $x \in I_k$ . The function  $f_x: I_k \rightarrow I_k$  is defined as follows:

For  $y = y_1 y_2 \dots y_k$ ,  $f_x(y) = G_{y_1 y_2 \dots y_k}(x)$ .

Define  $F_k = \{f_x\}_{x \in I_k}$  and  $F = \{F_k\}$ .

Note that a function in  $F_k$  needs not be one-to-one.

The reader may find it useful to picture a function  $f_x: I_k \rightarrow I_k$ , as a full binary tree of depth k with k-bit strings stored in the nodes and edges labelled 0 or 1. The k-bit string x will be stored in the root. If a k-bit string s is stored in an internal node, v, then  $G_0(s)$  is stored in v's left-son,  $v_l$ , and  $G_1(s)$  is stored in v's right-son,  $v_r$ . The edge  $(v, v_l)$  is labelled 0 and the edge  $(v, v_r)$  is labelled 1. The string  $f_x(y)$  is then stored in the leaf reachable from the root following the edge path labelled y.

It is easy to see that F satisfies properties (1) and (2) of poly-random collections. A proof that F satisfy also property (3) (pseudo-randomness) can be found in GGM (Main Theorem).

## GENERALIZATIONS

In some applications, we would like to have random functions from  $I_{P_3(k)} \rightarrow I_{P_4(k)}$ . E.g. in hashing we might want functions from  $I_k$  into  $I_{10}$ . We meet this need by introducing the collection  $F = \{F_k\}$  defined as follows: For  $x \in I_k$ ,  $f_x \in F_k$  is a function from  $I_{P_3(k)}$  into  $I_{P_4(k)}$  defined as follows. Let  $y = y_1 \dots y_{P_3(k)}$ . Define  $f_x(y) = I_{P_4(k)} G_{y_1 \dots y_{P_3(k)}}(x)$ , where  $I_{P_4(k)}(z)$  are the first  $P_4(k)$  bits output by G when fed input  $z \in I_k$ , where G is a CSPRB generator.

Such an F is also a poly-random collection: properties (1) and (2) trivially hold, and property (3) can be proved in a way similar to the proof of the Main Theorem in GGM.

## Appendix C: An(unsatisfactory)straightforward simulation of random functions

Assume one needs to be able to evaluate a function that looks as if it is randomly selected from  $H_k$ . One can argue that since he will only need to evaluate the function on polynomially many (in k) inputs, it is sufficient that he proceeds as follows:

Choose a CSPRB generator  $G$  and a random  $k$ -bit long seed  $s$ . This choice specifies a  $kt+1$ -bit long pseudo-random bit-sequence  $b_1, \dots, b_{kt+1}$  that can be used as securely as a truly random pad. Let  $x_1, \dots, x_j$  denote the chronologically ordered sequence of inputs on which the "random function"  $f$  has already been evaluated.

Assume now that  $f$  needs to be evaluated on an input  $y$ . If  $y = x_i$  for  $i=1 \dots j$ , then  $f(y)$  is set to be the  $j+1$ st block of  $k$  consecutive bits in the pseudo-random sequence. (I.e.  $f(y) = b_{k \cdot j+1} \dots b_{k \cdot j+k}$ ). Also,  $y$  is stored as the  $j+1$ st input (storing  $f(y)$  is optional). Otherwise, if  $y = x_i$  for some  $i$ ,  $f(y)$  is recomputed as the  $i$ th block of bits in the pseudo-random sequence (or is retrieved from memory).

Note that this procedure does not specify a function and thus does not meet the theoretical challenge. Furthermore, it wastes storage proportionally to the number of oracle queries (inputs on which the function has been evaluated). This is a strict lower bound! If the inputs are randomly chosen they cannot be compressed at all!

By means of a more clever use of CSPRB generators, our solution requires only  $k$  bits of storage. Thus it meets both the theoretical and the practical challenges.

#### ACKNOWLEDGEMENTS

We would like to thank Ron Rivest for suggesting the IFOF application.

## REFERENCES

- [AL] D. Angluin and D. Lichtenstein, Provable Security of Cryptosystems: a Survey, YaleU/DCS/TR-288, 1983
- [BBS] L. Blum, M. Blum and M. Shub, A simple secure pseudo random number generator, Advances in Cryptology: Proc. of CRYPTO-82, ed. D. Shaum, R. L. Rivest and A.T. Sherman. Plenum press 1983, pp 61-78.
- [BG] M. Blum and S. Goldwasser, An Efficient Probabilistic Public-Key Encryption Scheme Which Hides all Partial Information, preprint May 1984.
- [BM] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. COMPUT., Vol 13, No. 4, Nov. 1984.
- [B] G. Brassard, On computationally secure authentication tags requiring short secret shared keys, Advances in Cryptology: Proc. of CRYPTO-82, ed. D. Shaum, R.L. Rivest and A.T. Sherman. Plenum press 1983, pp 79-86.
- [CG] B. Chor and O. Goldreich, RSA Rabin least significant bits are  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  secure, MIT/LCS/TM-260, May 1984.
- [CW] J.L. Carter and M.N. Wegman, Universal classes of hash functions, Proc. 9th ACM Symp. on Theory of Computing, 1977, pp 106-112.
- [GGM] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, MIT/LCS/TM-244, November 1983.
- [GM] O. Goldreich and S. Micali, The weakest CSPRB generator implies the strongest one, in preparation.
- [GMT] S. Goldwasser, S. Micali and P. Tong. Why and how to establish a private code on a public network, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp 134-144.
- [RSA] R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Commun. ACM vol. 21, Feb. 1978, pp 120-126.
- [S] A. Shamir, On the Generation of Cryptographically Strong Pseudo-random Sequences, 8th International Colloquium on Automata, Languages, and Programming, Lect. Notes in Comp. Sci. 62, Springer Verlag, 1981.
- [Y] A.C. Yao, Theory and applications of trapdoor functions, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp 80-91.

# An *Efficient* Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information

*Manuel Blum*

Computer Science Department  
University of California at Berkeley

*Shafi Goldwasser \**

Laboratory for Computer Science  
Massachusetts Institute of Technology

## Abstract

This paper introduces the first probabilistic public-key encryption scheme which combines the following two properties:

- (1) **Perfect secrecy with respect to polynomial time eavesdroppers:** For all message spaces, no polynomial time bounded passive adversary who is tapping the lines, can compute any partial information about messages from their encodings, unless factoring composite integers is in probabilistic polynomial time.
- (2) **Efficiency:** It compares favorably with the deterministic RSA public-key cryptosystem in both encoding and decoding time and bandwidth expansion.

The security of the system we propose can also be based on the assumption that the RSA function is intractable, maintaining the same cost for encoding and decoding and the same data expansion. This implementation may have advantages in practice.

## 1. Introduction

Much attention has been devoted recently to investigating the security of cryptographic protocols, cryptographic encryption techniques, and digital signatures methods. Still, the most important problem in public-key cryptography remains how to encrypt both efficiently and securely.

**Key Words:** probabilistic encryption, partial information, integer factorization, passive adversaries, chosen cyphertext attack.

---

\* Supported in part by IBM Young Faculty Development Award Dated September 1984

It is customary to call an encryption system secure, when it is infeasible to recover the cleartext  $x$  from its encryption  $E(x)$ . However, this does not necessarily mean that it is infeasible to learn some partial information about  $x$  from  $E(x)$ , for  $x$ 's of interest. In fact, it has been pointed out in ([L],[GM]) that for any deterministic encryption scheme  $E$ , such as the RSA or Rabin's schemes, partial information about  $x$  can always be computed from  $E(x)$ . When  $E$  is used in cryptographic protocols, finding out this partial information can be detrimental to the security of the application, as has been shown for the protocols of "Mental Poker" in [SRA] and "Flipping Coins in Many Pockets" in [BD]. Moreover, whenever the messages to be sent are drawn from special message spaces (e.g. the messages can be expressed as known linear combinations of each other) the secrecy of the messages is in question ([B],[H]).

This issue has been rigorously defined and studied by Goldwasser and Micali in [GM]. They introduced the notion of a Public-Key Encryption scheme that is *polynomially secure*: a polynomial time adversary can not find one message  $m$  whose encodings he can distinguish from the encodings of a random message. An equivalent formulation of this security requirement is that after the eavesdropper sees an encrypted message passing in the network, his *a posteriori* probability of computing correctly which message is being sent remains roughly the same as his *a priori* probability of guessing which message is being sent before seeing the encrypted message. This implies that seeing the cyphertext does not help the adversary to compute or guess any function of the cleartext, better than he could have before seeing the cyphertext. Public key encryption schemes have been proposed that achieve this security criteria if the Quadratic Residuosity problem is intractable in [GM], and more generally if there exists any trapdoor function by Yao in [Y].

These schemes are probabilistic and are computed in a "bit-by-bit" fashion. In other words, every message has many possible encodings and every bit of a message is encrypted independently. Due to this last property, these schemes are highly inefficient. If  $k$  is the size of the security parameter (e.g. the size of the modulus in the RSA encryption function) then **each** bit is encoded individually by a  $k$ -bit long string in [GM] and even worse in [Y], resulting in at least a  $k$ -bit data expansion factor. Moreover, decoding a single bit in [GM] takes  $O(k^3)$  operations. In comparison, the RSA [RSA] or Rabin [Ra] encryption schemes, being deterministic block-ciphers, where a message is encrypted as a block of bits, transforms  $k$  bits of cleartext into  $k$  bits of cyphertext using  $O(k^3)$  operations.

This paper proposes a simple scheme which combines the polynomial security of the probabilistic schemes ([GM], [G], [Y]) and the efficiency of the deterministic schemes ([RSA] and [Ra]). Thus, with no loss of efficiency, this new scheme can be used in applications such as Mental Poker[GM2], Coin Flipping in a Distributed Network[ABCGM] or whenever messages to be sent come from a special message set and are dependent on one another in a publically known way.



Let  $k$  be the size of the security parameter. Then, the new scheme transforms an  $l$ -bit long cleartext into an  $(l+k)$ -bit long ciphertext, with a computational cost of  $O(\frac{lk^2}{\log k})$  for encryption and of  $O(k^3) + O(\frac{lk^2}{\log k})$  for decryption. When the length of the message  $l < k$  our scheme is at least as fast as either the RSA or Rabin schemes and when  $l > k$  our scheme is actually slightly faster. Most importantly, it is more secure. We prove that the scheme is polynomially secure, provided that factoring (or inverting RSA) is intractable.

## A High Level Description of Our System

Our work has gained from and extends the works by Goldwasser-Micali [GM], Blum-Micali [BM], Yao [Y], Blum-Blum-Shub [BBS], Goldwasser-Micali-Tong [GMT] and Chor-Goldreich [CG].

The notion of a cryptographically strong pseudo random bit (CSPRB) generator has been introduced by Blum and Micali in [BM] and extended by Yao in [Y]. Such a generator is a program which takes as input a  $k$ -bit random seed and produces as output a  $k'$ -bit sequence, where  $t > 0$  is fixed. The output sequences produced by a CSPRB generator are high quality pseudo-random sequences in the following sense: *if the  $k$  bit seed is totally unknown*, they cannot be distinguished from truly random sequences of the same length by any statistical test which runs in polynomial in  $k$  time.

The key idea in our method will be: to send an  $l$ -bit message  $m$ , send the exclusive-or of  $m$  with an  $l$ -bit output of a CSPRB generator on a  $k$ -bit random input seed  $S$  *along with a public-key encryption of  $S$* . Other implementations of this idea using different CSPRB generators and different types of encodings of  $S$  have been proposed by Blum, Blum and Shub in [BBS] and Goldwasser, Micali and Tong in [GMT], but they were not as efficient as our implementation and relied on different number theoretic assumptions.

We show a way to encrypt the seed  $S$  for which:

- (1) We prove that sending the encrypted seed  $S$  along with the exclusive-or of the message and the output of the CSPRB generator on input  $S$ , does not compromise the apparent "randomness" of the output of the CSPRB generator. (Yao [Y] proved that sequences outputted by CSPRB generators are polynomial time indistinguishable from truly random sequences, only when the seed is totally unknown).
- (2) The encoding and decoding of the seed  $S$  take at most  $O(k^3)$  steps and  $k$  bits of ciphertext, where  $k$  is the size of the seed.

## 2. Number Theoretic Background

Let  $|m|$  denote  $\log_2 m$  and  $Z_N^*$  denote the set of positive integers less than  $N$  which are relatively prime with  $N$ . Let  $(\frac{x}{N})$  denote the Jacobi symbol of  $x \in Z_N^* \bmod N$ . Recall that for

$x \in Z_N^*$  where the prime factorization of  $N$  is  $N = \prod_i p_i^{a_i}$

$$\left(\frac{x}{N}\right) = \prod_i \left(\frac{p_i^{a_i}}{N}\right)$$

and,

$$\left(\frac{x}{p_i}\right) = 1 \text{ if } x \text{ is a quadratic residue mod } N \text{ and } -1 \text{ otherwise}$$

Let  $N = pq$ , where  $p \equiv q \equiv 7 \pmod{8}$ . Set  $QR_N = \{a \mid \exists x, x^2 \equiv a \pmod{N}\}$ .

Then, any  $a \in QR_N$  has exactly two square roots with Jacobi symbol  $+1$  of the form  $x$  and  $-x$  in  $Z_N^*$  and exactly one square root with Jacobi symbol  $+1$  which is less than  $\frac{N}{2}$ . Let  $x$  be such a square root of  $a \in QR_N$ .

Define  $B(a, N)$  for  $a \in QR_N$  as follows:<sup>(1)</sup>

$$B(a, N) = \text{least significant bit of } x$$

Recent results by Chor and Goldreich [CG] imply that

**Lemma 1** [CG]: If there exists a probabilistic polynomial in  $|N|$  time algorithm that computes  $B(x, N)$  for a fraction  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  of  $x \in QR_N$  where  $N \in H$ , then there exists a probabilistic polynomial in  $|N|$  time algorithm for factoring  $N \in H$ .

They extend Lemma 1 as follows. Let

$$B^k(a, N) = k^{\text{th}} \text{ least significant bit of } x \text{ where } x^2 \equiv a \pmod{N}, x < \frac{N}{2}, \text{ and } \left(\frac{x}{N}\right) = +1.$$

**Lemma 2** [CG], [VV]: Let  $1 \leq j \leq \log \log N$ . If there exists a probabilistic polynomial in  $|N|$  time algorithm that on inputs  $N \in H$  and  $S \subseteq \{B^i(x, N) \mid 1 \leq i \leq \log \log N, i \neq j\}$ , guesses  $B^j(x, N)$  for a fraction greater than  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  of  $x \in QR_N$ , then there exists a probabilistic polynomial in  $|N|$  time algorithm for factoring  $N \in H$ .

### 3. The Encryption Scheme

Let the public file of user A contain a composite number  $N$  product of two primes  $p$  and  $q$ , both congruent to  $7 \pmod{8}$ . A keeps  $p$  and  $q$  secret.

Let the security parameter  $k = \log N$ . Define  $G(l, r, N)$  to be the  $l$ -bit boolean vector whose  $l-i+1$ st bit is  $B^s(r^{2^{k+1}}, N)$  where  $h = \left\lfloor \frac{l}{\log k} \right\rfloor$  and  $g = i - h \cdot \log k$  for  $1 \leq i \leq l$ .

(1) This predicate was first introduced in [GMT] and further analyzed in [BCS] and [VV].

Suppose any user  $B$  in the network wants to send an encoding of an  $l$ -bit message  $m \in M$  to user  $A$ .

**How to Encode  $m$ :**

1. Choose  $r \in \mathbb{Z}_N^*$  at random.
2. Compute  $G(l, r, N)$  using algorithm A below.
3. set  $f = r^{2^{h+1}} \bmod N$  where  $h = \left\lceil \frac{l}{\log k} \right\rceil$
4. Let the encryption of  $m$  be the pair  $(G(l, r, N) \oplus m, f)$ .

**Algorithm A**

*input:*  $l, r, N$  where  $k = \log N$ .

*output:*  $G(l, r, n)$

- 1) Let  $h = \max\{j \mid j \cdot \log k \geq l\}$ .
- 2) For  $j=0$  to  $h-1$  do  
     Let  $temp = r^{2^{j+1}} \bmod N$   
     For  $i = 1$  to  $\log k$ , let the  $l - (j \cdot \log k + i) + 1$  bit of  $G(l, r, n)$  be  $B^i(temp, N)$   
     (namely, the  $i^{th}$  least significant bit of  $temp$ ).

Note that the size of the encoding of an  $l$ -bit message  $m$  is  $l + k$ . Computing  $G(l, r, N)$  can be done in  $O(\frac{lk^2}{\log k})$  steps.

**How to Decode**

Let the pair  $(D, f)$  be a public-key encoding of an  $l$  bit message sent to user  $A$ . Then, user  $A$  who knows the factors of  $N$  does:

1. Let  $h = \frac{l}{\log k}$  2. compute  $r$  such that  $f^{2^{h+1}} = r \bmod N$  by performing the algorithm B below.
2. Compute  $G(l, r, N)$  using algorithm A above.
3. Let  $m = G(l, r, N) \oplus D$ .

The cost of decoding is  $O(k^3) + O(\frac{lk^2}{\log k})$ . This is faster than a previous  $O(k^4)$  of all previous probabilistic encoding schemes [GM, BBS, GMT]. We achieve this by exploiting the special properties of  $p$  and  $q$ . We show this in algorithm B below. The main idea is that instead of extracting square roots of  $f \bmod N$  at every step of the computation to retrieve  $G(l, r, N)$ , we first compute  $r$  in one exponentiation and then compute  $G(l, r, N)$  by algorithm A.

**Algorithm B:**

*Input:*  $p, q, h$  and  $r^{2^{h+1}} \bmod N$  where  $r \in \mathbb{QR}_N$  and  $k = \log N$ .

*Output:*  $r \in \mathbb{QR}_N$

*Running time:*  $O(k^3)$  operations.

Let  $p=8t+7$  and  $q=8l+7$  for some  $t$  and  $l$ .

1) Let  $u \equiv 2^{h+1}(t+1)^{h+1} \bmod p-1$

2) Let  $v \equiv 2^{h+1}(l+1)^{h+1} \bmod q-1$

(The above exponents  $u$  and  $v$  can be precomputed on inputs  $p, q$  and  $h+1$  alone)

3) Let  $a \equiv (r^{2^{h+1}})^u \bmod p$

4) Let  $b \equiv (r^{2^{h+1}})^v \bmod q$

5) Compute  $c \in \mathbb{Z}_N^*$  such that  $c \equiv a \bmod p$  and  $c \equiv b \bmod q$  using the Chinese Remainder theorem.

6) Output  $r = c$ .

Steps 1 and 2 above can be precomputed when the cryptosystem is set up. The cost of steps 3-4 are two modular exponentiations and one gcd computation of  $k$  bit numbers. The complexity of the naive algorithm for performing these operations is  $O(k^3)$ .

*Decoding Lemma:* The decoding algorithm is correct.

*Proof.* To compute square roots of  $a \in QR_p$ , note that  $(\frac{a}{p}) = a^{(t+3)/2} = +1$  implies  $(a^{2t+2})^2 = a \bmod p$ . Thus to compute the square root of  $a \bmod p$  which is a square itself, simply compute  $a^{2t+2} \bmod p$ . Finally, to compute the  $2^{h+1}$ -th root of  $a^{2^{h+1}} \bmod N$  which is a square, compute  $a^{(t+1)^{h+1}2^{h+1}} \bmod p$  and  $a^{(l+1)^{h+1}2^{h+1}} \bmod q$ , and take the Chinese remainder of the results mod  $pq$ . The exponents  $(t+1)^{h+1}2^{h+1} \bmod p-1$  and  $(l+1)^{h+1}2^{h+1} \bmod q-1$  do not depend on  $r$  and can be precomputed, knowing  $p, q$  and the length of the message.

## 4. Security Analysis

### 4.1 The Model

Let  $M$  be any message space. We think of the sender as a stochastic process which produces  $m \in M$  in accordance with a polynomial time computable system of probabilities. Thus every  $m \in M$  has an a priori probability of being sent. The adversary knows and can compute the probability distribution of the message space. When he intercepts the cyphertext he can calculate from it a set of a posteriori probabilities of the various messages which may have produced this cyphertext. Shannon defines a cypher to be *perfect secure* if: after the adversary intercepts the cyphertext, the a posteriori probability of the cyphertext representing some message must be the same as the a priori probability of the same message before interaction.

We adopt Shannon's perfect secrecy criteria to polynomial time bounded adversaries. See also [GM] and [Y].

Let  $k$  be the security parameter of our system. We think of a public-key encryption scheme as a probabilistic, polynomial time algorithm  $\Pi$  that receives  $k$  as input from each user in the system, and outputs a pair of polynomial in  $k$  time computable encryption and decryp-

tion algorithms  $E:M \rightarrow C$  and  $D:C \rightarrow M$  where  $C$  is the set of cyphertexts. The encryption algorithm  $E$  may be probabilistic. We let  $E(m)$  denote the set of possible encodings of  $m \in M$ . Note that when  $E = RSA$ ,  $E(m)$  is unique for any  $m \in M$ .

The measure of security we would like to enforce is the following.

Informally, no polynomial time passive adversary which can compute  $x \in E(m)$  for any  $m \in M$ , should be able to come up with even one message  $m$  whose encodings he can even distinguish from the encodings of a random message  $r \in M$ . This implies that seeing an encryption of a message does not help the adversary to compute or guess with any significant advantage any partial information about the message itself.

Formally, Let  $A$  be a polynomial time, probabilistic algorithm that takes as input a description of  $E$  and outputs a message  $m_{E,A} \in M$ . Let  $B$  be a polynomial time, probabilistic algorithm that takes as input  $x \in E(m)$  and outputs 1 or 0 (this is  $B$ 's guess as to whether  $m = m_{E,A}$  or not). Note that  $B$  may even know  $m_{E,A}$ . Let  $p_m$  denote the probability that  $B$  guesses 1 on  $x \in E(m)$  (the probability here is taken over the encodings  $x \in E(m)$  and  $B$ 's coin tosses).

*Definition:* We say that  $\Pi$  is *polynomially secure* if for all  $M$ , for all  $A, B$ , for all polynomials  $Q$ , for all sufficiently large  $k$ ,  $|p_{m_{A,E}} - p_r| < \frac{1}{Q(k)}$  for a random  $r \in M$  and random  $E$  generated by  $\Pi$ .

This notion of security was introduced by Goldwasser and Micali in [GM]. Another notion of security was later proposed by Yao[Y] using time-bounded information theory. Rackoff [R] showed the two notions equivalent, providing some evidence that polynomial security is the correct notion for security for a public-key cryptosystem.

## 4.2 Proof of Security

*Factoring Assumption (FA):* Let  $F_k^A$  denote be the fraction of  $k$ -bit integers factored by probabilistic, polynomial in  $k$  time algorithm  $A$ . Then, for all polynomials  $Q$ , for sufficiently large  $k$ ,  $F_k^A < \frac{1}{Q(k)}$ .

*Theorem:* The FA implies that the scheme described in section 3 is polynomially secure

*Idea of the proof:* Let  $\Pi$  be public-key encryption scheme that produces encryption algorithms of the form we proposed in section 3. Namely, a user in the system gives input  $k$  to  $\Pi$  and receives in return two  $k$ -bit prime numbers  $p$  and  $q$  such that  $p \equiv q \equiv 7 \pmod{8}$ . The user publicizes  $N = pq$  and keeps  $p$  and  $q$  secret. To send the user messages one encodes as we suggested in section 3. For the duration of the proof, let's denote this encryption algorithm by  $E_N$ , and the set of possible encryptions of message  $m$  by  $E_N(m)$ .

Now, suppose for contradiction, that for some pair  $A, B$  of polynomial-time probabilistic algorithms and for some polynomial  $Q$ , for infinitely many  $k$ , on a random  $E_N$  output by  $\Pi$  on  $k$

and a random  $r \in M$ , algorithm  $A$  on input  $E_N$  outputs an  $m_{A,E} \in M$  such that

$$|p_{m_{A,E}} - p_r| > \frac{1}{Q(k)} \quad (a)$$

Pick a  $k$  for which (a) holds. Assume without loss of generality that for all  $m \in M$ ,  $|m| = l$  and  $l < P(k)$  for polynomial  $P$ . Pick a random  $N$  (product of two  $k$ -bit primes  $p$  and  $q$  such that  $p \equiv q \equiv 7 \pmod{8}$ ) whose factors you don't know. Input  $N$  to algorithm  $A$  to receive  $m_{A,E} \in M$ . Now, let's define two types of experiments.

Type (1): Pick  $x \in QR_N$  at random and an  $l$ -bit random  $R$ . Let  $h = \max\{j \mid j \log k \geq l\}$ . Feed algorithm  $B$ ,  $x^{2^{h+1}} \bmod N$  and  $R \oplus G(l, x, N)$  (i.e. a random member of  $E_n(R)$ ). The probability that  $B$  answers 1 here is  $p_R$ .

Type (2): Pick  $x \in QR_N$  at random. Feed algorithm  $B$ ,  $x^{2^{h+1}} \bmod N$  and  $m_{A,E} \oplus G(l, x, N)$  (i.e. a random member of  $E_n(m_{A,E})$ ). The probability that  $B$  answers 1 here is  $p_{m_{A,E}}$ .

By assumption,  $|p_{m_{A,E}} - p_R| > \frac{1}{Q(k)}$ . Without loss of generality, let  $p_{m_{A,E}} - p_R > \frac{1}{Q(k)}$ . By the weak law of large numbers, in polynomial in  $Q(k)$  number of type (1) and type (2) experiments we can estimate  $p_{m_{A,E}}$  and  $p_R$ .

We now use a combination of the proof techniques of Goldwasser and Micali in [GM] and Yao in [Y]. Let the  $i$ -type experiment be defined as follows. Feed algorithm  $B$ ,  $r^{2^{h+1}} \bmod N$  and  $G(l, r, N) + m_i$  where  $m_i$  consists of the concatenation of the first  $i$  bits of  $m_{A,E}$  and  $l-i$  random bits (i.e. feed  $B$  a random member of  $E_N(m_i)$ ). Let  $p_i$  be the probability that during the  $i$ -type experiment  $B$  outputs 1. (this probability is taken over  $B$ 's coin tosses, choices of  $r$  and the  $l-i$  random bits). There must exist an  $1 \leq i \leq l$  such that  $p_{i+1} - p_i > 1/lQ(k)$  and it can be found by running a polynomial in  $lQ(k)$  number of  $i$ -type experiments and  $i+1$ -type experiments for all  $1 \leq i \leq l$ . Say we found such an  $i$ . There are two cases to look at:  $i \leq \log k$  and  $i > \log k$ . The first case yields a method for predicting  $B^i(x, N)$  with probability greater than  $\frac{1}{lQ(k)}$  on inputs  $N$  and  $\{B^j(x, N) \mid 2 \leq j \leq l\}$ , which by lemma 2 implies

a probabilistic factoring algorithm for  $N \in H$ . This contradicts the factoring assumption. In the second case  $i > \log k$ . The rest of the proof deals with this case. Pick an  $a$  in  $Z_N^*$  such that  $(\frac{a}{N}) = -1$ . Let  $G$  consist of the concatenation of  $G(i, a^2, N)$  with  $l-i$  random bits. Let  $h' = \max\{j \mid j \log k \geq l\}$ . Feed algorithm  $B$  as input,  $a^{2^{h'+1}} \bmod N$  and  $G \oplus m^0$  where  $m^0$  consists of the concatenation of the first  $i$  bits of  $m_{A,E}$ , 0, and  $l-i-1$  random bits. Denote  $B$ 's output by  $b_0$ . (Note that this is an  $i$ -type experiment). Now feed  $B$  with  $a^{2^{h'+1}} \bmod N$  and  $G + m^1$  where  $m^1$  consists of the concatenation of first  $i$  bits of  $m_{A,E}$ , 1, and  $l-i-1$  random bits. Denote  $B$ 's output by  $b_1$ . (Note that this is an  $i+1$ -type experiment). Let  $j = \log k$  and let  $b_{i+1}$  denote the  $i+1$ 'st bit of  $m_{A,E}$ . If  $b_0 = b_1$  then choose at random  $c \in \{0, 1\}$  and predict

that  $B^j(a, N) = c$ , otherwise let  $c = 0 \cdot b_0 + 1 \cdot b_1$  and predict  $B^j(a, N) = b_{i+1} \oplus c$ . The probability of predicting  $B^j(a, N)$  correctly according to this rule is greater than  $\frac{1}{2} + \frac{1}{Q(k)l}$ . This yields by Lemma 2 a probabilistic time algorithm for factoring  $N \in H$ , which again contradicts the factoring assumption.

## 5. Efficiency Analysis and Comparison

Let  $k$  be the security parameter (i.e. the size of the composite number in the public file). To send an  $k$  bit message  $m$  where  $k = \log N$  using RSA requires  $O(k^3)$  operations to encode and decode and  $k$ -bit long encryption. In our scheme encoding requires  $O(\frac{k^3}{\log k})$  operations, decoding requires  $O(k^3) + O(\frac{k^3}{\log k})$  operations and the encryption of a  $k$ -bit message is  $2k$ -bit long.

Note that the time bounds for RSA encoding and decoding have been calculated for  $\text{RSA}(x) = x^s \bmod N$  where  $1 < s < \varphi(N)$  is picked at random as proposed in the original RSA paper. It has been suggested to let  $s=3$ , then encoding is of  $k^2$  complexity, while decoding still remains of  $k^3$  complexity. However, Blum in [B] and Hastad in [H] point out a problem arising with  $s=3$  (or any  $s < \log k$ ): if the same message is sent encrypted to  $3s$  (respectively) separate people in the network each owning his own  $N_i$ , then an adversary tapping the lines can decode the message.

## 6. Other models of adversaries

The security analysis performed in section 4 was done with respect to passive adversaries. However, the scheme described above is not secure against more powerful than passive adversaries such as adversaries which can perform chosen cypher text attacks (CCA). In this attack, the adversary may have temporary access to the decoding equipment, after which he tries to decode. No public key encryption scheme has been proved secure against such an attack even under the assumption that certain number theoretic problems are intractable.

On the other hand, no effective way of inverting the RSA function using a CCA on the RSA public-key-cryptosystem is known. We can modify our scheme to achieve the same security against CCA, as the deterministic RSA cryptosystem. We do this by modifying our system to be based on the assumption that the RSA function is intractable. This implementation maintains the same cost of encoding and decoding, same data expansion, and the same security against partial information attacks as our factoring based scheme. In addition it is as secure against CCA as the deterministic RSA system. We briefly describe this implementation in section 7.

Note that what we are interested in is an encryption scheme which is *1-pass*. That is, to send a message to user A we need only look up his public file, compute the encryption of our message and send it, and there is no need of further interaction with A. Solving the problem of

public-key encryption which is secure against chosen-ciphertext attacks using a 2-pass system (where to send a secret message to A two levels of interaction are allowed) is much easier and can be achieved.

## 7. Implementation of the Scheme based on RSA intractability assumption

Let  $N$  denote the modulus in the RSA scheme. That is  $N = pq$  where  $p$  and  $q$  are primes of the same size. Let the public file of user A contain such a composite number  $N$  whose factors  $p$  and  $q$  only A knows. Let  $s_i$  denote the inverse of  $3^i \bmod \varphi(N)$  for  $1 < i < \varphi(N)$ . Let  $k = \log N$ .

Define  $G(l, r, N)$  of section 4 to be an  $l$ -bit vector whose  $l-i+1$ th bit is the  $i - \log k \left\lfloor \frac{i}{\log k} \right\rfloor$  least significant bit of  $r^{3^j} \bmod N$  where  $j = \left\lfloor \frac{i}{\log k} \right\rfloor$ . Note that computing  $G(l, r, N)$  can be done without knowing the factors of  $N$  in  $O\left(\frac{lk^2}{\log k}\right)$  time.

To encode  $m$  where  $|m| = l$ : pick  $r \in \mathbb{Z}_N^*$  at random, compute  $G(l, r, N)$ , let  $h = \left\lceil \frac{l}{\log k} \right\rceil$  and  $f = r^{3^{h+1}} \bmod N$ , let the encoding of  $m$  be the pair  $(G(l, r, N) \oplus m, f)$ .

To decode  $(D, f)$  where  $|D| = l$ , recall that  $s_{h+1} 3^{h+1} \bmod \varphi(N) \equiv 1$ . Compute  $r = f^{s_{h+1}} \bmod N$ , compute  $G(l, r, N)$ , and let  $m = G(l, r, N) \oplus D$ .

## 8. Remarks and Open problems

This paper presented a probabilistic encryption scheme which is secure against all partial information attacks in presence of passive adversaries, provided factoring is hard, whose cost of encoding and decoding is fast, and has constant factor data expansion. An interesting open question remains:

Given  $x^2 \bmod N$ , are  $1/2$  of the bits of  $x$  such that  $x < \frac{N}{2}$  and  $\left(\frac{x}{N}\right) = +1$  as hard to compute as  $x$ ? If so, then one may build an extremely efficient encryption scheme which requires only 2 multiplications to encode, and is secure against all partial information attacks.

## Acknowledgements

We thank Ron Rivest, Dana Angluin, Don Coppersmith, Oded Goldreich and Silvio Micali for their helpful comments.

## References

- [ABCGM] Awerbach, Blum, Chor, Goldwasser, Micali, *A Provably Fair Coin Toss in A Byzantine Network*, Submitted to PODC 1985.
- [CG] Chor, Goldreich, *RSA/Rabin Bits are  $1/2 + \frac{1}{\text{poly}(|N|)}$  secure*, Proc. of Crypto 84, Santa Barbara.



- [B] M. Blum, private communication .
- [BBS] L. Blum, M. Blum and M. Shub, *A simple secure pseudo random number generator*, Advances in Cryptology: Proc. of CRYPTO-82, ed. D. Chaum, R.L. Rivest and A.T. Sherman. Plenum press 1983, pp 61-78.
- [BCS] Ben-Or, Chor, Shamir, *On the Security of RSA Bits*, Proceedings of 15th ACM symposium on Theory of Computation, April 1983, pp. 421-430
- [BD] A. Broder, and D.Dolev, *On Flipping Coins in Many Pockets*, 25th IEEE FOCS, 1984.
- [BM] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp 112-117
- [DH] Diffie and Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory.
- [GM] S. Goldwasser and S. Micali, *Probabilistic Encryption*, JCSS 28(2), 1984. References
- [GM2] Goldwasser and Micali, *Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information*, 1982 14th STOC.
- [GMT] S. Goldwasser, S. Micali and P. Tong, *Why and how to establish a private code on a public network*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp 134-144
- [GMR] S. Goldwasser, S. Micali and R. Rivest, *Probabilistic Signature Secure Against Chosen Cyphertext Attack*, In Preparation.
- [H] J. Hastad, *On Using RSA with Low Exponent in A Public Key Network*, In Preparation.
- [L] D. Lipton, *How to Cheat in Mental Poker*.
- [Ra] M. Rabin, *Digital Signatures as Intractable as Factorization*.
- [RSA] R. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Commun. ACM, vol. 21, Feb. 1978, pp 120-126
- [Y] A.C. Yao, *Theory and applications of trapdoor functions*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp 80-91.
- [Sh] C. Shannon, *A Mathematical Theory of Cryptography* , 1945.
- [VV] V. Vazirani, U. Vazirani, *Trapdoor Pseudo-Random Number Generators, with Applications to Protocol Design* , 1983.

# RSA/Rabin least significant bits are $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$ secure

(Extended Abstract)

Benny Chor \* Oded Goldreich \*\*

Laboratory for Computer Science  
Massachusetts Institute of Technology  
Cambridge, Massachusetts 02139

**Abstract** — We prove that RSA least significant bit is  $\frac{1}{2} + \frac{1}{\log^c N}$  secure, for any constant  $c$  (where  $N$  is the RSA modulus). This means that an adversary, given the ciphertext, cannot guess the least significant bit of the plaintext with probability better than  $\frac{1}{2} + \frac{1}{\log^c N}$ , unless he can break RSA.

Our proof technique is strong enough to give, with slight modifications, the following related results:

- (1) The  $\log \log N$  least significant bits are simultaneously  $\frac{1}{2} + \frac{1}{\log^c N}$  secure.
- (2) The above also holds for Rabin's encryption function.

Our results imply that Rabin/RSA encryption can be **directly** used for pseudo random bits generation, provided that factoring/inverting RSA is hard.

## 1. INTRODUCTION

Given a secure public key cryptosystem [7], it is hard to recover the plaintext  $x$  from its encryption,  $E(x)$ . However, this does not necessarily mean that a cryptanalyst cannot gain some partial information about  $x$  without actually computing it. The ability to derive partial information can render a cryptosystem useless in specific applications (e.g. mental poker [18],[13],[11]). For example, even a moderate ability of guessing the least significant bit of the plaintext may be a threat to security.

In the current state of knowledge we are unable to prove even the existence of secure public key cryptosystems. However, under reasonable assumptions on the computational complexity of certain problems, secure public key cryptosystems do exist and can be explicitly constructed. One of the most fascinating questions regarding those systems is "*what partial information about the plaintext is hard to extract from the ciphertext?*"

This question was rigorously defined and studied, with respect to probabilistic encryption, by Goldwasser and Micali [11]. They constructed a public key cryptosystem which leaks no partial

\* Research supported in part by the National Science Foundation under Grant MCS-8006938.

\*\* Research supported in part by a Weizmann Postdoctoral Fellowship.

information. However, their system encrypts messages by expanding each plaintext bit into a ciphertext block, making it undesirable from a practical point of view.

The RSA [16] is the most widely known public key cryptosystem, and probably the first one which will be used in practice. It has been an open problem to demonstrate a predicate,  $P(\cdot)$ , such that having any advantage in guessing  $P(x)$  given the encryption of  $x$ , is as hard as inverting RSA.

In this paper, we show that RSA least significant bit is  $\frac{1}{2} + \epsilon$  secure for any polynomial fraction  $\epsilon$  ( $\epsilon^{-1} = O(\log^c N)$ ), where  $N$  is the RSA modulus. With a small modification, our proof technique also allows us to show that  $\log \log N$  of RSA bits are **simultaneously**  $\frac{1}{2} + \frac{1}{\log^{\frac{1}{2}} N}$  secure. I.e., if RSA is indeed secure, then no heuristic which runs in polynomial time can get information about any function of these plaintext bits, given the ciphertext. Hence these bits provide instances of secure partial information for RSA.

Our results have important implications for generating sequences of cryptographically strong pseudo-random bits. RSA encryption  $E$  can be **directly** used for generating such sequences by starting from a random seed  $s$  and iterating  $E$  on it.

Slightly modifying our proof techniques, we also prove the same strong bit security for Rabin public key scheme [15]. This implies a fast and "direct" pseudo-random bits generator which is as hard to crack (distinguish its outputs from truly random strings) as factoring. Important consequences follow also w.r.t the probabilistic encryption scheme of Goldwasser & Micali [11] (see section 8.2).

**Organization of the paper :** In section 2 we formally define the question of security for RSA least significant bit and cover previously known results. In section 3 we sketch the proof of Ben-Or, Chor & Shamir result, and in section 4 - its improvement by Schnorr & Alexi. These two investigations are the basis for our work, which is described in section 5. Section 6 extends our proof to other RSA bits and section 7 - to bits in Rabin's scheme. Section 8 contains concluding remarks on the applications of our results for the **direct** construction of pseudo-random bit generators and probabilistic encryption schemes.

## 2. PROBLEM DEFINITION AND PREVIOUS RESULTS

The RSA encryption function is operating in the message space  $Z_N$ , where  $N = pq$  is the product of two large primes (which are kept secret). The encryption of  $x$  is  $E_N(x) = x^e \pmod{N}$ , where  $e$  is relatively prime to  $\varphi(N) = (p-1)(q-1)$ . For  $0 \leq x < N$ ,  $L(x)$  denotes the least significant bit in the binary representation of  $x$ .

Let  $O_N$  be an oracle which, given  $E_N(x)$ , outputs a guess for  $L(x)$  (this guess might depend on a random coin used by  $O_N$ ). Let  $p(N)$  be a function from integers into the interval  $[\frac{1}{2}, 1]$ . We say that  $O_N$  is a  $p(N)$ -oracle if the probability that the oracle is correct, given  $E_N(x)$  as its input, is  $p(N)$  (the probability space is that of all  $x \in Z_N$  with uniform distribution and -if  $O_N$  uses a random coin- also of all 0-1 sequences of coin tosses with uniform distribution).

We say that RSA least significant bit is  $p(N)$ -secure if there is a probabilistic polynomial time algorithm which inverts  $E_N$ , using queries of any  $p(N)$ -oracle  $O_N$ . Since an unbiased coin can be used as an  $\frac{1}{2}$ -oracle, the best possible security result can be  $\frac{1}{2} + \epsilon$  security for any  $\epsilon^{-1} = \text{poly}(\log N)$  ( $\frac{1}{2}$  security means RSA is breakable). These notions originate from Blum & Micali's work [5], where they have been stated w.r.t the discrete exponentiation function.

Goldwasser, Micali and Tong [12] showed that the least significant bit is as hard to compute as inverting the RSA. Furthermore, they showed that it is  $(1 - \frac{1}{\log N})$ -secure.

Ben-Or, Chor and Shamir [1] showed a  $\frac{3}{4} + \epsilon$  security ( $\epsilon^{-1} = \text{poly}(\log N)$ ). They presented an algorithm which inverts the RSA by carrying out a gcd calculation on two multiples of the ciphertext and using any  $(\frac{3}{4} + \epsilon)$ -oracle. Sampling the oracle they amplified its  $\frac{3}{4} + \epsilon$  advantage to "almost certainty", for a polynomial fraction of the message space.

Vazirani and Vazirani [19] improved the result using a novel oracle-sampling technique. They proved that their modification is guaranteed to succeed when given access to any 0.732-oracle.

Goldreich [9] used a better combinatorial analysis to show that the Vazirani & Vazirani modification inverts even when given access to a 0.725-oracle. He also pointed out some limitations of the Vazirani & Vazirani and similar proof techniques.

Schnorr and Alexi [17] introduced a conceptual change in the way the oracle is used. This enabled them to greatly improve the result showing that the least significant bit is  $(\frac{1}{2} + \epsilon)$ -secure for any constant  $\epsilon > 0$ . Their result still leaves a gap towards the optimal  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  security.

### 3. A SKETCH OF BEN-OR CHOR AND SHAMIR ALGORITHMIC PROCEDURE

#### The essence of the Inverting Algorithm:

Given an encrypted message,  $E_N(x)$ , the plaintext  $x$  is reconstructed by performing a gcd algorithm on two small multiples of it (small means in the interval  $[\frac{-N\epsilon}{2}, \frac{N\epsilon}{2}] \pmod{N}$ ). A special binary variant is used for the gcd algorithm. To operate, this variant needs to know the parity of the absolute value of  $O(\log^2 N)$  small multiples of the plaintext. Thus, it is provided with a *subroutine* that determines the parity of these multiples.

#### Determining Parity using an Oracle which may err:

The *subroutine* determines the parity of a small multiple  $d = kx$ , of the plaintext  $x$ , by using an  $p(N)$ -oracle for RSA's l.s.b as follows. It picks a random  $r$  and asks the oracle for the least significant bit of both  $rx$  and  $rx + d$ , by feeding it in turn with  $E_N(rx) = E_N(r)E_N(x)$  and  $E_N((r+k)x) = E_N(r+k)E_N(x)$ . The oracle's answers are processed according to the following observation. Since  $d = kx$  is "small", with very high probability no wrap around 0 occurs when  $d$  is added to  $rx$ . Then, the parity of  $|d|$  is equal to 0 if the least significant bits of  $rx$  and  $rx + d$  are identical; and equal to 1 otherwise. This is repeated many times; every repetition (instance) is called a  $d$ -measurement. Note that the outcome of a  $d$ -measurement is correct if the oracle was correct on both  $rx$  and  $rx + d$  (the outcome is also correct if the oracle was wrong on both queries, but this fact is not used in [1]).

#### (Trivial) Measurement Analysis:

A  $d$ -measurement is correct with probability at least  $1 - 2(1 - p) = 2p - 1$ .

(This suffices if  $p = \frac{3}{4} + \epsilon$ .)

### 4. A SKETCH OF SCHNORR AND ALEXI IMPROVEMENT: $\frac{1}{2} + \epsilon$ FOR ANY CONSTANT $\epsilon$

Schnorr & Alexi [17] improvement is based on trying all possibilities for the least significant bit of  $L = \theta(\log \log N)$  random, independent positions  $w_i = r_i x$  and using these positions as "end points" in all measurements for the  $O(\log^2 N)$   $d$ 's of the binary gcd algorithm. This way the oracle is queried only about one end-point of each measurement and the error is caused by

single position queries rather than by pairs of positions. This enables the error probability per a single measurement to be approximately the oracle's error, rather than twice this magnitude as in Ben-Or, Chor & Shamir. Using the fact that the  $L$  positions are independent, Chernoff bound implies that the error probability in deciding the parity of  $d$  by the majority of  $d$ -measurements is  $2^{-\Omega(L\epsilon^2)} < \frac{1}{\log^3 N}$  (here  $\epsilon$  is a constant). This guarantees that the accumulated error probability in deciding the parity of all  $O(\log^2 N)$   $d$ 's in the modified binary *gcd* algorithm is  $< \frac{1}{2}$ , small enough to put the algorithm in random polynomial time.

Note that the running time of Schnorr & Alexi's algorithm is exponential in  $L$ . On the other hand, the probabilistic analysis requires that  $L = \Omega(\frac{\log \log N}{\epsilon^2})$ . Thus,  $\epsilon$  can not be replaced by any function which tends to 0 with  $N \rightarrow \infty$ .

## 5. OUR MAIN RESULT

In this section we prove that RSA least significant bit is  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  secure.

Let  $\mathcal{O}_N$  be an oracle for RSA least significant bit whose error probability is  $\frac{1}{2} - \epsilon$ , where  $\epsilon^{-1} \leq \log^c N$ .

Instead of picking  $\theta(\log \log N)$  random **independent** positions, we generate  $L = \theta(\log^{2c+3} N)$  random positions which are only **pairwise independent**, such that we know (with very high probability) the least significant bit of each. As in Schnorr and Alexi's work, we query the oracle only about one end-point of each measurement and use the same "decision by majority" idea. Since the positions are not independent, Chernoff bound cannot be used in our case. However, since the points are pairwise independent, Chebyshev's inequality still holds. It gives an  $O(\frac{1}{L\epsilon^2})$  upper bound on the error probability. With  $L$  being so large, this error is sufficiently small.

### Generating $L$ "random" positions knowing their least significant bits

We generate  $L$  positions by picking two random independent variables  $y, z \in Z_N$  and trying all possibilities for their least significant bits and location in one of the intervals  $[i\frac{N}{L^3}, (i+1)\frac{N}{L^3})$ ,  $0 \leq i < L^3$ . There are  $(2L^3)^2$  possibilities altogether, and exactly one of them is correct. Let us now assume that we are dealing with the correct choice, i.e. both least significant bit and approximate magnitude of  $y, z$  are known. The positions we'll look at are  $w_i = y + iz \pmod{N}$  for  $i = 1, 2, \dots, L$ . Notice that  $w_i$  is a random element in  $Z_N$  with uniform probability distribution. Since the location of both  $y$  and  $z$  are known up to  $\frac{N}{L^3}$ , the location of  $w_i = y + iz$  is known up to  $\frac{N}{L^3} + i\frac{N}{L^3} < \frac{2N}{L^3}$ . The probability of  $w_i$  to be within an interval of length  $\frac{2N}{L^3}$  containing 0  $\pmod{N}$  is exactly  $\frac{2}{L^3}$ . If  $w_i$  is **not** in such interval, then its least significant bit is determined by  $i$  and the least significant bits of  $x$  and  $y$ . Therefore we get

$$\Pr(\text{least significant bit of } w_i \text{ is unknown}) \leq \frac{2}{L^3}.$$

### Determining parity using the generated positions and the oracle

Let  $d \in Z_N$  be any fixed "small" number (one of those generated by the *gcd* procedure). In order to determine the parity of  $|d|$ , we'll query the oracle about all points of the form  $w_i + d$ , XOR the answers with the (known) least significant bits of the corresponding  $w_i$ , and take the

majority.<sup>1</sup> Using Chebyshev's inequality, we'll get a bound for the probability that the majority of the oracle's answers will be biased to the wrong direction.

### Error analysis :

Suppose  $d \in Z_n$  is any "small" number (in the interval  $[-\frac{N\epsilon}{2}, \frac{N\epsilon}{2}]$ ). For a random  $r \in Z_N$ , the probability that a wrap around 0 (mod  $N$ ) occurs when  $d$  is added to  $r$  is no greater than  $\frac{\epsilon}{2}$ . Hence if  $|d|$  is even, the probability that  $O_N$ , on input  $E_N(r+d)$ , gives the same answer as the (true) least significant bit of  $r$  is at least  $\frac{1}{2} + \epsilon - \frac{\epsilon}{2} = \frac{1}{2} + \frac{\epsilon}{2}$ . Similarly, if  $|d|$  is odd, then with probability at least  $\frac{1}{2} + \frac{\epsilon}{2}$ ,  $O_N$  answer to the least significant bit of  $r+d$  is different than the least significant bit of  $r$ . By the above discussion, we get

$$Pr(w_i + d \text{ did not wrap around and } O_N \text{ is correct on it}) \geq \frac{1}{2} + \frac{\epsilon}{2}, \text{ for every } i.$$

Define

$$\zeta_i = \begin{cases} 0 & \text{if } w_i + d \text{ did not wrap around and } O_N \text{ is correct on it and } w_i \text{ l.s.b. is known} \\ 1 & \text{otherwise} \end{cases}$$

Hence

$$\begin{aligned} Pr(\zeta_i = 0) &\geq Pr(w_i + d \text{ did not wrap around and } O_N \text{ is correct on it}) \\ &\quad - Pr(w_i \text{ least significant bit is unknown}) \\ &\geq \frac{1}{2} + \frac{\epsilon}{2} - \frac{2}{L^2} \\ &> \frac{1}{2} + \frac{\epsilon}{4} \quad (\text{for } L > \sqrt{\frac{8}{\epsilon}}). \end{aligned}$$

Therefore,  $Exp(\zeta_i) = Pr(\zeta_i = 1) < \frac{1}{2} - \frac{\epsilon}{4}$  and

$$Var(\zeta_i) = Exp(\zeta_i^2) - Exp^2(\zeta_i) = Exp(\zeta_i) - Exp^2(\zeta_i) = Exp(\zeta_i)(1 - Exp(\zeta_i)) < \frac{1}{4}.$$

Since  $Exp(\zeta_i) < \frac{1}{2} - \frac{\epsilon}{4}$ , we get

$$Pr\left(\frac{1}{L} \sum_{i=1}^L \zeta_i \geq \frac{1}{2}\right) \leq Pr\left(\left|\frac{1}{L} \sum_{i=1}^L \zeta_i - Exp(\zeta_i)\right| \geq \frac{\epsilon}{4}\right).$$

We can apply Chebyshev's inequality (see Feller [8, p. 219]) and get,

$$Pr\left(\left|\frac{1}{L} \sum_{i=1}^L \zeta_i - Exp(\zeta_i)\right| \geq \frac{\epsilon}{4}\right) \leq \frac{Var(\frac{1}{L} \sum_{i=1}^L \zeta_i)}{(\epsilon/4)^2}.$$

Since  $y$  and  $z$  are independent random variables and  $y + iz$ ,  $y + jz$  are linearly independent for  $i \neq j$ , then  $w_i$  and  $w_j$  are also independent random variables for any  $i \neq j$ . Therefore, for any

<sup>1</sup>Notice that this decision procedure is exactly the one employed in Ben-Or, Chor & Shamir. The crucial difference is that they had to use the oracle's answer to find  $w_i$ 's least significant bit, while we know it beforehand (with overwhelming probability).

$i \neq j$ ,  $\zeta_i$  and  $\zeta_j$  are also independent random variables with identical distribution. (Whenever the same function is applied to two independent random variables, the two results are independent random variables). Let  $\bar{\zeta}_i = \zeta_i - \text{Exp}(\zeta_i)$ . By pairwise independence,  $\text{Exp}(\bar{\zeta}_i \cdot \bar{\zeta}_j) = \text{Exp}(\bar{\zeta}_i) \cdot \text{Exp}(\bar{\zeta}_j)$ . Hence,

$$\begin{aligned} \text{Var}\left(\frac{1}{L} \sum_{i=1}^L \zeta_i\right) &= \text{Exp}\left(\left(\frac{1}{L} \sum_{i=1}^L \bar{\zeta}_i\right)^2\right) = \frac{1}{L^2} \sum_{i=1}^L \sum_{j=1}^L \text{Exp}(\bar{\zeta}_i \cdot \bar{\zeta}_j) \\ &= \frac{1}{L^2} \left( \sum_{i=1}^L \text{Exp}(\bar{\zeta}_i^2) + \sum_{1 \leq i \neq j \leq L} \text{Exp}(\bar{\zeta}_i) \text{Exp}(\bar{\zeta}_j) \right) = \frac{1}{L^2} \cdot L \cdot \text{Exp}(\bar{\zeta}_1^2) < \frac{1}{4L}. \end{aligned}$$

Thus the probability that  $\frac{1}{L} \sum_{i=1}^L \zeta_i \geq \frac{1}{2}$  is smaller than  $\frac{4}{L\epsilon^2}$ . But  $\Pr\left(\frac{1}{L} \sum_{i=1}^L \zeta_i \geq \frac{1}{2}\right)$  is exactly the error probability for a single  $d$ . We query at most  $\log^2 N$   $d$ 's in the course of the  $\text{gcd}$  computation and thus the error probability (for one binary  $\text{gcd}$ ) is bounded by

$$\log^2 N \cdot \Pr(\text{error for a single } d).$$

Taking  $L = \log^{2c+3} N$ , the overall error probability is bounded from above by

$$\log^2 N \cdot \frac{4}{L\epsilon^2} \leq \frac{4\log^2 N}{\log^{2c+3} N (\log^{2c} N)^{-1}} = \frac{4}{\log N}.$$

Hence we can recover the original message in random polynomial time, as desired. This implies **Theorem 1**: RSA least significant bit is  $(\frac{1}{2} + \frac{1}{\log^c N})$ -secure, for any constant  $c$ .

## 6. OTHER RSA BITS

Our proof technique easily extends to provide strong security results for several other RSA bits. In particular the following holds:

### Theorem 2:

a) Let  $I \subseteq [0, N]$  be an interval of length  $N/2$ . The  $I$  bit of  $x$  is the characteristic function of  $I$  (i.e. 1 if  $x \in I$  and 0 otherwise). This bit is  $(\frac{1}{2} + \frac{1}{\log^c N})$ -secure.

b) Let  $k = O(\log \log N)$ . The  $k$ -th bit in the binary expansion of the plaintext is  $\frac{1}{2} + \frac{1}{\log^c N}$  secure.

c) Let  $k = O(\log \log N)$ . The plaintext's  $k$  least significant bits are **simultaneously** secure. I.e., even if all least significant bits  $x_{k-1}, \dots, x_2, x_1$  are given together with  $E_N(x)$ , still  $x_k$  is  $(\frac{1}{2} + \frac{1}{\log^c N})$ -secure.<sup>1</sup>

d) All bits in the binary expansion of  $x$  (except maybe the  $\log \log N$  most significant ones) are  $(\frac{3}{4} + \frac{1}{\log^c N})$ -secure. At least half of them are  $(\frac{11}{16} + \frac{1}{\log^c N})$ -secure.

### Proof sketch :

(a) and (d) follow from Theorem 1, by reductions due to Ben-Or, Chor and Shamir [1].

<sup>1</sup>Equivalently, given  $E_N(x)$  distinguishing between  $x_k \dots x_2 x_1$  and a randomly selected string of length  $k$  is as hard as inverting the RSA. This equivalence is due to Yao [21].

b) First note that using our proof technique, it is possible to guess all  $k$  least significant bits of  $y$  and  $z$ . This determines all  $k$  least significant bits of each  $w_i$ .

Apply the  $gcd$  procedure to two small multiples of the plaintext, the greatest common divisor of which is  $2^k$ . This way all  $d$ 's in the  $gcd$  calculation will have zeros in all  $k-1$  least significant bits. Replace all reference to the least significant bit, in the inverting algorithm (presented in section 5), by references to the  $k$ -th bit. Note that this time we have access to an oracle to the  $k$ -th bit.

(This method of transforming certain inverting algorithms which use an oracle for the 1-st bit into inverting algorithms which use an oracle for the  $k$ -th bit originates from Vazirani and Vazirani [19].)

c) Going through the proof of Theorem 2(b), notice that when querying the oracle about the  $k$ -bit of  $w_i + d$  we can give it the  $k-1$  previous bits of  $w_i + d$ . (The latter are equal to the  $k-1$  least significant bits of  $w_i$ , which we know!)

(Vazirani and Vazirani [20] had previously shown that, certain inverting algorithms which use a  $p(N)$  oracle for RSA least significant bit, can be transformed into inverting algorithms which use a  $p(N)$  oracle for predicting  $x_k$  (given  $x_{k-1}, \dots, x_1$ ). It turns out that the inverting algorithm of section 5 falls into the above category; this yields an alternative (but much harder) way of proving Theorem 2(c).)

## 7. BITS EQUIVALENT TO FACTORING IN RABIN'S ENCRYPTION FUNCTION

### 7.1 Previous Results

The Rabin encryption function is operating in the message space  $Z_N$ , where  $N = pq$  is the product of two large primes (which are kept secret). The encryption of  $x$  is  $E_N(x) = x^2 \pmod{N}$ . The ciphertext space is  $Q_N = \{y | \exists x \ y \equiv x^2 \pmod{N}\}$ . Rabin [15] has shown that extracting square roots ("inverting  $E_N$ ") is polynomially equivalent to factoring.

Note that the function  $E_N$  defined above is 4 to 1 rather than being 1 to 1 (as is the case in the RSA). Blum [2] has pointed out that if  $p \equiv q \equiv 3 \pmod{4}$  then  $E_N$  induces a permutation over  $Q_N$ . These  $N$ 's will hereby be called *Blum integers*. Goldwasser, Micali and Tong [12] have presented a predicate the evaluation of which is as hard as factoring. Specifically, they showed that if  $p \equiv 3 \pmod{4}$  and  $p \equiv q \pmod{8}$  then factoring  $N$  is polynomially reducible to guessing their predicate with success probability  $1 - \frac{1}{\log N}$ .

Ben-Or, Chor and Shamir [1] considered the same predicate. Using a modification of their RSA techniques they showed  $\frac{3}{4} + \epsilon$  security for this predicate. Their modification requires that  $N$  be a Blum integer and furthermore that there exists a small odd number  $l$  ( $l = O(\log^c N)$ ) with  $(\frac{l}{N}) = -1$ . Its correctness proof makes use of non-elementary number theory.

### 7.2 Our Result

We transform our RSA security result into a similar result for the Rabin encryption function. Our transformation is simpler than the one used in [1], and its correctness proof is elementary. Furthermore, it holds for any Blum integer.

Let  $N$  be a Blum integer,  $S_N = \{x | 0 \leq x < \frac{N}{2}\}$  and  $M_N = \{x | 0 \leq x < \frac{N}{2} \ \& \ (\frac{x}{N}) = 1\}$ .



Redefining  $E_N$  for  $x \in M_N$  as

$$E_N(x) = \begin{cases} x^2 \pmod{N} & \text{if } x^2 \pmod{N} < \frac{N}{2} \\ -x^2 \pmod{N} & \text{otherwise} \end{cases}$$

makes  $E_N$  a 1-1 mapping from  $M_N$  onto itself, without losing the intractability result of Rabin. i.e. factoring  $N$  is polynomially reducible to inverting  $E_N$ . Let  $L(x)$  be the least significant bit of  $x$ .

The main idea in the reduction (as in the RSA case) is to pick  $L$  positions  $w_i \in S_N$  which are uniformly distributed in  $S_N$  and pairwise independent, such that their least significant bits are known. Some difficulties arise, but they can be taken care off (see [6]). We get

**Theorem 3:** The least significant bit for the modified Rabin encryption function is  $(\frac{1}{2} + \frac{1}{\log^c N})$ -secure, for any constant  $c$ .

**Corollary:** Factoring a Blum integer,  $N$ , is polynomially reducible to guessing  $L(x)$  with success probability  $\frac{1}{2} + \frac{1}{\log^c N}$  when given  $E_N(x)$ , for  $x \in M_N$ .

The proofs from the previous section about simultaneous security of  $\log \log N$  least significant bits and of bit intervals (for intervals of length  $\frac{N}{4}$  out of the  $\frac{N}{2}$  long interval containing  $M_N$ ) hold here just as well, thus all these bits are also  $\frac{1}{2} + \frac{1}{\log^c N}$  secure.

## 8. APPLICATIONS

### 8.1 Direct Construction of Pseudo-Random Bit Generators

A pseudo-random bits generator is a device which "expands randomness". Given a truly random bit sequence  $s$  (the seed), it expands it to a longer pseudo-random sequence. The question of "how random" this pseudo-random sequence is depends on what exact definition of randomness we are after. A strong requirement is that the expended sequence will pass all polynomial time statistical tests, namely given a pseudo-random and a truly random sequences of equal length, no probabilistic polynomial time algorithm can tell which is which with better than 50-50 success (this definition was proposed by Yao [21], who also showed it is equivalent to some other natural definitions like unpredictability).

Blum and Micali were the first to construct such strong pseudo-random generators. Their construction combines two results:

- If  $g: M \rightarrow M$  is a 1-1 one way function, and  $B_N(x)$  is  $\frac{1}{2} + \epsilon$  secure bit for  $g$  (where  $\epsilon =$  any polynomial fraction), then starting with a random  $s \in M$ , the sequence obtained by iterating  $g$  and outputting  $b_i = B(g^i(s))$  for each iteration is pseudo random (in the sense that each of its bits can not be predicted better than 50-50, from the previous ones).
- Demonstrating that a specific bit is  $\frac{1}{2} + \epsilon$  secure for the discrete exponentiation function.

We say that a generator is *direct w.r.t* the (underlying) one way function  $g$  if it produces at least one bit per one iteration of  $g$ . We say that a generator is *strong w.r.t* an (assumed) intractable problem,  $P$ , if distinguishing its output from truly random sequences is as hard as solving  $P$ . Notice that both the Blum & Micali generator and the Long & Wigderson generator<sup>1</sup> ([14]) are

<sup>1</sup>Long & Wigderson's generator produces  $\log \log p$  bits per each iteration of the discrete exponentiation  $(\text{mod } p)$  function. This is due to their proof that this function has  $\log \log p$  simultaneously hard bits.

direct w.r.t discrete exponentiation and strong w.r.t discrete log.

Another direct generator was constructed by Blum, Blum and Shub [3]. Their generator is direct w.r.t squaring modulo a composite number and was proven strong w.r.t deciding quadratic residuosity.

Yao [21] made some generalizations to the Blum & Micali result. He showed that having a 1-1 one way function  $f$  is enough and it is not necessary to have a specific secure bit. The main idea is that if  $f$  is one way then some bits must be secure (even though not necessarily  $\frac{1}{2} + \epsilon$  secure). Picking a polynomial number of random seeds  $\{s_{j,k}\}$ , we get one strongly pseudo-random bit  $b_i$  by computing

$$\bigoplus_j \bigoplus_k B_j(f^i(s_{j,k}))$$

( $B_j(s)$  is the  $j$ -th bit of  $s$   $\bigoplus$  is the one bit XOR of the result.)

Yao XORing trick works for any 1-1 one way function,  $f$ , but the generators achieved that way are not direct w.r.t  $f$  - to produce one bit, many applications of  $f$  are needed. For further details on Yao's XORing trick and its proof consult Goldwasser [10].

All previously known results about the cryptographic security of Rabin/RSA scheme (including Schnorr & Alexi result) do not suffice for constructing generators which are strong w.r.t factoring/inverting the RSA and direct w.r.t Rabin/RSA encryption function.

With  $\frac{1}{2} + \frac{1}{\text{poly} \log N}$  security, we can finally get generators which are direct w.r.t Rabin/RSA encryption function and strong w.r.t factoring/inverting RSA. Each of the bits whose  $\frac{1}{2} + \frac{1}{\text{poly} \log N}$  security is proven can be used as the "hard bit" the generator outputs. As a matter of fact, with the stronger result that all  $\log \log N$  least significant bits are simultaneously  $\frac{1}{2} + \frac{1}{\text{poly} \log N}$  secure, we can get  $\log \log N$  random bits per one application of the encryption function. Since the encryption in Rabin scheme is just one squaring and one subtraction, we get a very fast generator, whose security is equivalent to factoring a Blum integer<sup>1</sup>.

Using our techniques, Vazirani and Vazirani [20] have pointed out that the Blum, Blum and Shub [3] generator is strong also w.r.t. factoring Blum integers.

## 8.2 Direct Construction of Probabilistic Encryption Schemes

Observation, similar to the ones of section 8.1, apply to the probabilistic encryption scheme suggested by Goldwasser and Micali [11]. Using our result we introduce the first direct probabilistic encryption equivalent to factoring/inverting RSA. However, this implementation still has the bandwidth expansion drawback; the plaintext is expanded by a factor of  $O(\frac{\log N}{\log \log N})$ .

Recently, Blum and Goldwasser [4] used our result to introduce a new implementation of probabilistic encryption, equivalent to factoring, in which the plaintext is only expanded by a constant factor. Goldwasser's scheme is approximately as efficient as the RSA while provably leaking no partial information, provided that factoring is intractable.

## ACKNOWLEDGMENTS

We would like to thank Michael Ben-Or, Shafi Goldwasser, Silvio Micali and Ron Rivest for

<sup>1</sup>A composite number  $N = pq$ , such that  $p$  and  $q$  are both primes congruent to 3 modulo 4

very helpful discussions and useful ideas.

Oded Goldreich would like to thank Dassi Levi for her unique existence.

## REFERENCES

- [1] Ben-Or, M., Chor, B., and Shamir, A., "On the Cryptographic Security of Single RSA Bits", *15th ACM Symp. on Theory of Computation*, April 1983, pp. 421-430.
- [2] Blum, M., "Coin Flipping by Telephone", *IEEE Spring COMCON*, 1982.
- [3] Blum, L., Blum, M., and Shub, M., "Comparison of Two Pseudo-Random Number Generators", *Advances in Cryptology: Proceedings of Crypto82*, Chaum, D., et al. eds., Plenum Press, 1983, pp. 61-79.
- [4] Blum, M., and Goldwasser, S., "An Efficient Probabilistic PKCS as Secure as Factoring", these proceedings.
- [5] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", to appear in the *SIAM Jour. on Computing*.
- [6] Chor, B., and Goldreich, O., "RSA/Rabin least significant bits are  $\frac{1}{2} + \frac{1}{\text{poly}(\log N)}$  Secure", MIT/LCS/TM-260, May 1984.
- [7] Diffie, W., and Hellman, M.E., "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.
- [8] Feller, W., *An Introduction to Probability Theory and its Applications*, John Wiley & Sons Inc., Vol. I, (1962).
- [9] Goldreich, O., "On the Number of Close-and-Equal Pairs of Bits in a String (with Implications on the Security of RSA's L.s.b.)", MIT/LCS/TM-256, March 1984.
- [10] Goldwasser, S., "Probabilistic Encryption: Theory and Applications", Ph.D. Thesis, Berkeley, 1984.
- [11] Goldwasser, S., and Micali, S., "Probabilistic Encryption", *Jour. of Computer and System Science*, Vol. 28, No. 2, 1984, pp. 270-299.
- [12] Goldwasser, S., Micali, S., and Tong, P., "Why and How to Establish a Private Code on a Public Network", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, November 1982, pp. 134-144.
- [13] Lipton, R., "How to Cheat at Mental Poker", Proceeding of the AMS short course on Cryptology, January 1981.
- [14] Long, D.L., and Wigderson, A., "How Discreet is Discrete Log?", *15th ACM Symp. on Theory of Computation*, April 1983, pp. 413-420. A better version is in preparation.
- [15] Rabin, M.O., "Digital Signatures and Public Key Functions as Intractable as Factorization", MIT/LCS/TR-212, 1979.
- [16] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signature and Public Key Cryptosystems", *Comm. of the ACM*, Vol. 21, February 1978, pp. 120-126.

- [17] Schnorr, C.P. and Alexi, W., "RSA bits are  $0.5 + \epsilon$  Secure", presented at EuroCrypt84, Paris, April 1984.
- [18] Shamir, A., Rivest, R.L., Adleman, L., "Mental Poker", MIT/LCS/TM-125, February 1979.
- [19] Vazirani, U.V., and Vazirani, V.V., "RSA Bits are  $.732 + \epsilon$  Secure", *Advances in Cryptology: Proceedings of Crypto83*, Chaum, D. ed, Plenum Press, 1984, pp. 369-375.
- [20] Vazirani, U.V., and Vazirani, V.V., "Efficient and Secure Pseudo-Random Number Generation", these proceedings.
- [21] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.

INFORMATION THEORY WITHOUT THE FINITENESS ASSUMPTION, I:  
CRYPTOSYSTEMS AS GROUP-THEORETIC OBJECTS

G. R. Blakley

Department of Mathematics  
Texas A&M University  
College Station, Texas 77843-3368

1. INTRODUCTION

This paper gives a definition of cryptosystem in terms of confusion, diffusion and replacement. This definition lends itself to infinite, as well as finite, structures, and the notion of group appears to play an essential role in it. We offer three theses for discussion. The first is that all known cryptosystems fit the definition. The second is that (Shannon) confusion amounts to left composition of a cryptographic relation with a message and left action of a cryptographic relation on a message, as well as that (Shannon) diffusion amounts to left composition of a message with a cryptographic relation and left action of a message on a cryptographic relation. The third is that what Shannon calls mixing cannot occur unless a certain type of "nonassociativity", or at least lack of adherence to some algebraic laws, is present in the description of a cryptosystem in accordance with this definition.

The three theses are supported by examples below. If the first cannot be readily falsified, it would be interesting to express every cryptosystem -- as well as the known cryptanalytic attacks on it -- in the style of this paper. If the second cannot, it might be appropriate to use it as a precise definition of the notions of confusion and diffusion. If the third cannot, it might be a jumping-off point for a mathematical exploration of mixing.

The approach of this paper can suggest new cryptosystems. It describes finite cryptosystems and infinite cryptosystems (such as analog speech scramblers) with equal facility. It organizes and simplifies the current variety of descriptions of cryptosystems. It is purely formal and has no place for mechanical or electrical

notions, such as lug, pin, rotor, box, etc. It can, however, describe the workings of crypto boxes which use such devices.

## 2. MESSAGES, CONFUSION, DIFFUSION AND REPLACEMENT

We define a message to be a map (i.e. function)

$$m: P/E \rightarrow A/L$$

where  $P$  and  $A$  are groups [PA66, p. 79],  $E$  is a normal subgroup [PA66, p. 110] of  $P$ , and  $L$  is a normal subgroup of  $A$ . We speak of  $P/E$  as being the set of character positions, and of  $A/L$  as being the alphabet. In other symbolism, a message  $m$  is a member of the cardinal power [MA67, p. 15]  $(A/L)^{P/E}$ . We use the algol arrow notation for exponents, and so we write instead

$$m \in (A/L) \uparrow P/E.$$

A confusion operation  $s$  is a (binary) relation [PA66, p. 15] on  $A/L$ , i.e. a subset of  $A/L \times A/L$ . A diffusion operation  $t$  is a (binary) relation on  $P/E$ . A replacement operation is a (binary) relation  $r$  on  $(A/L) \uparrow P/E$ . Usually a replacement operation is a function

$$r: (A/L) \uparrow P/E \rightarrow (A/L) \uparrow P/E,$$

i.e. is a map which turns a message into another message. Our first thesis states that all known cryptosystems are families of cryptographic keys, and that every cryptographic key is a succession of confusion, diffusion and replacement operations or messages. Thus this paper actually moves away from the generality of the "family of maps" definition which dominates the literature [DE82, p. 7; DI79, p. 398; KO81, p. 28] at present.

For example a simple substitution cryptosystem key  $s$  is the simplest kind of confusion operation. It is a permutation of  $A/L$ , i.e. a member of the set  $\text{SYM}(A/L)$  consisting [KO81, p. 65] of all permutations of  $A/L$ . To encrypt a message

$$m \in (A/L) \uparrow P/E$$

by means of a simple substitution key

$$s \in \text{SYM}(A/L) \subseteq (A/L) \uparrow A/L$$

one forms the ordinary composite [PA66, p. 63] function

$$s \circ m \in (A/L) \uparrow P/E$$

defined by setting

$$(s \circ m)(p) = s(m(p))$$

for every  $p \in P/E$ . As an example we will consider the plaintext message  $m = \text{SUBSTITUTION}$ . Let us take

$P = \mathbb{Z}$ , the integers under addition

$E = 12\mathbb{Z} = \{\dots, -12, 0, 12, 24, 36, \dots\}$

$A = \mathbb{Z}$

$L = 26\mathbb{Z}$

Thus we can view SUBSTITUTION as a function

$$m: \mathbb{Z}/12\mathbb{Z} \rightarrow \mathbb{Z}/26\mathbb{Z}$$

where we source-code by making the identification

$$1 \leftrightarrow A$$

$$2 \leftrightarrow B$$

.

.

.

$$25 \leftrightarrow Y$$

$$26 = 0 \leftrightarrow Z$$

In this case we have, for example,

$m(1) = S \quad (= 19, \text{really}),$   
 $m(2) = U \quad (= 21),$   
 $m(3) = B \quad (= 2),$   
 $\vdots$   
 $m(11) = O \quad (= 15),$   
 $m(12) = N \quad (= 14).$

If  $s$  is the Caesar [KO81, pp. 69-72] cipher

$$s: \mathbb{Z}/26\mathbb{Z} \rightarrow \mathbb{Z}/26\mathbb{Z}$$

defined by setting

$$s(\zeta) = \zeta + 2 \quad (\text{modulo } 26)$$

i.e.

$$A \rightarrow C$$

$$B \rightarrow D$$

.

.

.

$$X \rightarrow Z$$

$$Y \rightarrow A$$

$$Z \rightarrow B.$$

then  $s \circ m$  is the message

$$s \circ m(1) = s(m(1)) = s(19) = 21 = U$$

$$s \circ m(2) = s(m(2)) = s(21) = 23 = W$$

$$s \circ m(3) = s(m(3)) = s(2) = 4 = D$$

.

.

.

$$s \circ m(11) = s(m(11)) = s(15) = 17 = Q$$

$$s \circ m(12) = s(m(12)) = s(14) = 16 = P$$

in other words the plaintext message SUBSTITUTION is replaced by the



cryptext message UWDUVKVWVKQP under the Caesar cryptosystem key "move two places down the alphabet."

To exemplify diffusion at the simplest level we use a transposition cipher key  $t$  on the plaintext message TRANSPOSITIONS. To model this we can choose to take:

$$P = \mathbb{Z} ;$$

$$E = O\mathbb{Z} = \{0\}, \text{ the small trivial subgroup;}$$

$$A = \mathbb{Z} ;$$

$$L = 27\mathbb{Z} .$$

Here we have chosen to identify letters, plus the blank symbol, of the latin alphabet with members of  $\mathbb{Z}/27\mathbb{Z}$  as follows:

$$27 = 0 \leftrightarrow \text{blank}$$

$$1 \leftrightarrow A$$

$$2 \leftrightarrow B$$

.

.

.

$$25 \leftrightarrow Y$$

$$26 \leftrightarrow Z .$$

Then the message

$$m: \mathbb{Z}/O\mathbb{Z} \rightarrow \mathbb{Z}/27\mathbb{Z}$$

amounts to the function

$$m: \mathbb{Z} \rightarrow \mathbb{Z}/27\mathbb{Z}$$

defined by setting

$$m(\zeta) = 0 \quad \text{if } \zeta \leq 0$$

$$m(\lambda) = 0 \quad \text{if } \lambda \geq 15$$

$$m(1) = T \quad (= 20, \text{ really}),$$

$$m(2) = R \quad (= 18),$$

$$m(3) = A \quad (= 1),$$

.

.

.

$$m(13) = N \quad (= 14)$$

$$m(14) = S \quad (= 19) .$$

We can choose a transposition  $t$  which turns blocks of 7 letters around, i.e. we can choose  $t$  such that

$$t(7\delta + \beta) = 7\delta + (8-\beta)$$

for any integer  $\delta$ , any positive integer  $\beta \leq 7$ . To encrypt the plaintext message  $m$ , i.e. the 14-letter block,

### TRANSPOSITIONS

preceded and followed by infinitely many blanks, using the transposition cryptosystem key  $t$  we form the ciphertext message  $m \circ t$ . Evidently

$$(m \circ t)(\zeta) = m(t(\zeta)) = 0$$

if  $\zeta \leq 0$  (for then  $t(\zeta) \leq 0$ ) and

$$(m \circ t)(\lambda) = m(t(\lambda)) = 0$$

if  $\lambda \geq 15$  (for then  $t(\lambda) \geq 15$ ). But

$$(m \circ t)(1) = m(t(1)) = m(7) = 0$$

$$(m \circ t)(2) = m(t(2)) = m(6) = P$$

.

$$(m \circ t)(6) = m(t(6)) = m(2) = R$$

$$(m \circ t)(7) = m(t(7)) = m(1) = T$$

$$(m \circ t)(8) = m(t(8)) = m(14) = S$$

$$(m \circ t)(9) = m(t(9)) = m(13) = N$$

.

$$(m \circ t)(13) = m(t(13)) = m(9) = I$$

$$(m \circ t)(14) = m(t(14)) = m(8) = S$$

The ciphertext  $m \circ t$  is thus the 14-letter block

## OPSNARTSNOITIS

preceded and followed by infinitely many blanks. We note, in passing, that the definition of  $t$  is most naturally framed in terms of the cosets of the subgroup  $7Z$  of the group  $Z = P$  of message positions. We will return to this theme later.

To sum up, our second thesis is that confusion (as Shannon [SH49] used the term) is applied to a message  $m$  by forming its left composite  $s \circ m$  with a relation  $s$  on its codomain [MA67, p. 4]. If, in particular,  $s$  is a permutation we have substitution. Continuing the second thesis, we assert that Shannon diffusion is applied to a message  $m$  by forming its right composite  $m \circ t$  with a relation  $t$  on its domain [MA67, p. 4]. If, in particular,  $t$  is a permutation we have transposition. In either case the natural group operation on the structure in question may be utilized to produce the needed permutations (or, more generally, functions or, most generally, relations).

So far we have treated the only two ways you can form composites involving a message  $m$ , on the right and on the left. There remains the possibility of regarding  $m$  itself as a domain point, and applying a function to it. This is the idea behind replacement. Replacement is a function  $r$  whose domain consists of messages and whose codomain also does. One example, though an imperfect one, of replacement is a code book.

This paper will concentrate largely on finite non-Shannon cryptosystems, i.e. collections of keys which act on finite alphabets, and which are not based on the idea of many successive applications of confusion, diffusion and replacement. The DES, an archetypal Shannon "roll the dough and fold it" [SH49] cryptosystem, will be treated in a later paper.

### 3. MONALPHABETS AND CAESARS

The composite  $q \circ s$  of two simple (i.e. monalphabetic) substitution cipher keys

$$q: A/L \rightarrow A/L$$

$$s: A/L \rightarrow A/L$$

is itself a simple substitution cipher key

$$q \circ s: A/L \rightarrow A/L$$

which encrypts a message

$$m: P/E \rightarrow A/L$$

by the rule which defines

$$(q \circ s) \circ m: P/E \rightarrow A/L$$

by setting

$$((q \circ s) \circ m)(p) = (q \circ s)(m(p)) = q(s(m(p)))$$

for every  $P/E$ . The associativity,  $(q \circ s) \circ m = q \circ (s \circ m)$ , of function composition is what Hellman calls the closure property. It means that one cannot achieve greater strength through mixing when one merely follows one substitution by another. Since  $\text{SYM}(A/L)$  is a group it follows that the collection  $\text{MON}[A/L]$  of all monalphabetic substitution cipher keys on an alphabet  $A/L$  is a group isomorphic to  $\text{SYM}(A/L)$ . The collection  $\text{LCAE}[A/L]$  of all left Caesar cipher keys on  $A/L$  is defined as follows. For each  $\alpha \in A/L$  define

$$\lambda[\alpha]: A/L \rightarrow A/L$$

by setting

$$\lambda[\alpha](\beta) = \alpha \# \beta,$$

where  $\#$  is the group operation of  $A/L$ . To encrypt a message

$m: P/E \rightarrow A/L$  form the composite function

$$\lambda[\alpha] \circ m: P/E \rightarrow A/L$$

defined by setting

$$\begin{aligned} (\lambda[\alpha] \circ m)(\beta) &= \lambda[\alpha](m(\beta)) \\ &= \alpha \# m(\beta) \end{aligned}$$

for every  $\beta \in P/E$ . The collection  $\text{RCAE}[A/L]$  of right Caesar cipher keys is defined analogously. If  $A/L$  is abelian then  $\text{LCAE}[A/L] = \text{RCAE}[A/L] = \text{CAE}[A/L]$ , the set of all two-sided Caesar cipher keys on  $A/L$ . It is obvious from the proof of Cayley's theorem [PA66, pp. 120-121] that  $\text{LCAE}[A/L]$  (under function composition) is isomorphic to  $A/L$ . Similarly  $\text{RCAE}[A/L]$  is isomorphic to  $A/L$ .

A heuristic principle suggests itself here. If there are only about as many keys in a simple substitution cipher as there are

letters in the alphabet, you may be dealing with a Caesar cipher. We shall, in accordance with this heuristic principle, see that the Pohlig-Hellman and the Rivest-Shamir-Adleman number-theoretic cryptosystems are very Caesar-like.

#### 4. POLYALPHABETS AND ONE-TIME PADS

We now give the group-theoretic formulation of classical polyalphabetic ciphers. Let us again use source coding to replace the latin alphabet by the set  $\{0, 1, 2, \dots, 25\} \subseteq \mathbb{Z}$

$$\begin{array}{rcl} A & \mapsto & 1 \\ B & \mapsto & 2 \\ & & \vdots \\ Z & \mapsto & 26 = 0 \end{array}$$

and let us agree to regard it not merely as a set [HA60, pp. 1-3], but as an additive [MA67, p. 71] abelian [MA67, pp. 71-77] group,  $\mathbb{Z}/26\mathbb{Z} = C_{26} = \mathbb{Z}_{26}$  [MA67, pp. 129-132]. Here, as often below, we allow ourselves to indulge in the common abuse of language which uses equality (=) where isomorphism [MA67, pp. 56-57] ( $\cong$ ) is meant. A polyalphabetic (n alphabets) cipher key is determined by a family [HA60, p. 34]

$$s: \mathbb{Z}/n\mathbb{Z} \rightarrow \text{SYM}(\mathbb{Z}/26\mathbb{Z})$$

of permutations [MA67, p. 72] of the "alphabet"  $\mathbb{Z}/26\mathbb{Z}$ . The family is indexed by the cosets [MA67, p. 51] (let us agree to call them by the coset leader names  $1, 2, \dots, n$ ) within  $\mathbb{Z}$  (considered as an additive abelian group) modulo the subgroup [MA67, p. 84]  $n\mathbb{Z}$ .

From the plaintext message

$$m: \mathbb{Z} \rightarrow \mathbb{Z}/26\mathbb{Z}$$

and the n-alphabetic cipher key

$$s: \mathbb{Z}/n\mathbb{Z} \rightarrow \text{SYM}(\mathbb{Z}/26\mathbb{Z})$$

we form the ciphertext message

$$y: Z \rightarrow Z/26Z$$

according to the rule

$$y(e+\lambda) = s_{\lambda}(m(e+\lambda))$$

for every member  $e$  of the subgroup  $E = nZ$ , every coset leader name  $\lambda \in \Lambda = \{1, 2, \dots, n\}$ . There is no harm in making the identification

$$\Lambda = Z/nZ$$

as long as you stick to a particular set of coset leaders (i.e. a particular set of names of cosets). In a strict mathematical sense  $\Lambda$  is the range of a choice function [HA60, p. 60]

$$f: Z/nZ \rightarrow Z.$$

Such a function  $f$  has the property that  $f(Q) \in Q$  for every coset  $Q = j + nZ = \{j + n\zeta: \zeta \in Z\}$  of  $nZ$  in  $Z$ .

To exemplify this definition in a 3-alphabet substitution case, take the message

#### POLYALPHABET

i.e.

$$m: Z/12Z \rightarrow Z/27Z$$

such that

$$m(1) = 16 = P$$

$$m(2) = 15 = O$$

$$m(3) = 12 = L$$

$$m(4) = 25 = Y$$

$$m(5) = 1 = A$$

$$m(6) = 12 = L$$

$$m(7) = 16 = P$$

$$m(8) = 8 = H$$

$$m(9) = 1 = A$$

$$m(10) = 2 = B$$

$$m(11) = 5 = E$$

$$m(12) = 20 = T$$

and take

$$s: \mathbb{Z}/3\mathbb{Z} \rightarrow \text{SYM}(\mathbb{Z}/27\mathbb{Z})$$

defined by setting

$$s_0(\alpha + 0) = \alpha + 2$$

$$s_1(\alpha + 1) = \alpha + 3$$

$$s_2(\alpha + 2) = \alpha + 5$$

for each  $\alpha \in 3\mathbb{Z}$ , where we have taken

$$\Lambda = \mathbb{Z}/3\mathbb{Z} = \{0, 1, 2\}.$$

Thus

$$s_1(m(1)) = s_1(16) = 16 + 3 = 19 = S$$

$$s_2(m(2)) = s_2(15) = 15 + 5 = 20 = T$$

$$s_0(m(3)) = s_0(12) = 12 + 2 = 14 = N$$

$$s_1(m(4)) = s_1(25) = 25 + 3 = 2 = B$$

$$s_2(m(5)) = s_2(1) = 1 + 5 = 6 = F$$

$$s_0(m(6)) = s_0(12) = 12 + 2 = 14 = N$$

$$s_1(m(7)) = s_1(16) = 16 + 3 = 19 = S$$

$$s_2(m(8)) = s_2(8) = 8 + 5 = 13 = M$$

$$s_0(m(9)) = s_0(1) = 1 + 2 = 3 = C$$

$$s_1(m(10)) = s_1(2) = 2 + 3 = 5 = E$$

$$s_2(m(11)) = s_2(5) = 5 + 5 = 10 = J$$

$$s_0(m(12)) = s_0(20) = 20 + 2 = 22 = V$$

So the 3-alphabetic encryption of

POLYALPHABET

in this key  $s$  is

STNBFNSMCEJV .

Suppose that  $n = 1$ . Then  $nZ = 1Z = Z$  is the large trivial subgroup and

$$Z/nZ = Z/Z = \{0\} = 0Z$$

(up to isomorphism). In this case the key  $s$  is a one-permutation family, i.e. a monalphabetic substitution cipher key. If  $n = 0$  then

$$nZ = 0Z = \{0\}$$

is the small trivial subgroup and

$$Z/nZ = Z/0Z = Z/\{0\} = Z$$

(up to isomorphism). In this case the key  $s: Z \rightarrow \text{SYM}(Z/26Z)$  is an infinite family of permutations of the "alphabet"  $Z/26Z$ , one for each plaintext message letter. This is a one-time substitution (somewhat fancier than the classical bitwise one-time pad [K081, p. 135] which uses  $Z/2Z$  rather than  $Z/26Z$  as its alphabet).

Evidently the underlying structure which embraces monalphabets (including Caesars), polyalphabets, and one-time pads is

$$m: P \rightarrow A/L$$

$$s: P/E \rightarrow \text{SYM}(A/L)$$

$$y: P \rightarrow A/L$$

where  $P$  is a group with normal [MA67, p. 106] subgroup  $E$ , where  $A$  is a group with normal subgroup  $L$ , where  $\text{SYM}(A/L)$  is the set (it is in an obvious and natural sense a group, of course) of permutations of  $A/L$ , where  $m: P \rightarrow A/L$  is an arbitrary message, where  $\#$  is the group operation in  $P$ , and where

$$y(e \# \lambda) = s_{\lambda}(m(e \# \lambda))$$

for every  $e \in E$ , every coset [MA67, pp. 101-103] leader  $\lambda \in \Lambda = P/E$  (equality being used where isomorphism is meant). Such a structure is called a (polyalphabetic) substitution cipher key. You can use right cosets instead of left cosets, with the obvious changes. Most



classical cryptosystems are based on additive [MA67, p. 71] abelian groups, so cosets are two-sided.

The composite of two substitution cipher keys

$$s: P/E \rightarrow \text{SYM}(A/L)$$

$$\bar{s}: P/\bar{E} \rightarrow \text{SYM}(A/L)$$

is a substitution cipher key.

$$u: P/(E \cap \bar{E}) \rightarrow \text{SYM}(A/L)$$

In the commonest case we are dealing with residue classes [PA66, p. 53] of integers:

$$P/E = \mathbb{Z}/n\mathbb{Z}.$$

$$P/\bar{E} = \mathbb{Z}/\bar{n}\mathbb{Z}.$$

We let  $m = \text{lcm}(n, \bar{n})$ , the least common multiple [PA66, p. 44] of  $n$  and  $\bar{n}$  and we find that

$$n\mathbb{Z} \cap \bar{n}\mathbb{Z} = m\mathbb{Z}$$

Thus the composite of a 12-alphabetic substitution key on the alphabet  $A/L$  and a 42-alphabetic substitution key on  $A/L$  (in either order) is an 84-alphabetic substitution key on  $A/L$ .

The composite of a simple substitution cipher key with an  $n$ -alphabetic substitution cipher key is  $n$ -alphabetic. The composite of any substitution cipher key with a one-time pad is a one-time pad. The collection of all substitution cipher keys on an alphabet  $A/L$  forms a group.

## 5. INFINITE SUBSTITUTION CIPHERS AND TORSION

The classical Vernam/Mauborgne one-time pad using a two-member alphabet can be described as

$$m: \mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$$

$$s: \mathbb{Z}/0\mathbb{Z} \rightarrow \text{SYM}(\mathbb{Z}/2\mathbb{Z})$$

$$y(\lambda) = y(0+\lambda) = s_{\lambda}(m(0+\lambda)) = s_{\lambda}(m(\lambda))$$

for every  $\lambda \in \mathbb{Z}$ . Since  $0\mathbb{Z}$  is the inverse limit (i.e. projective limit [MA71, pp. 68-72], in this case the intersection [HA60, pp. 14-18]) of the members of the sequence

$$\mathbb{Z} = \{\mathbb{Z}, 2\mathbb{Z}, 3\mathbb{Z}, \dots\}$$

we can say that  $\mathbb{Z} = \mathbb{Z}/0\mathbb{Z}$  is a sort of limit of  $\mathbb{Z}/\mathbb{Z}, \mathbb{Z}/2\mathbb{Z}, \mathbb{Z}/3\mathbb{Z}, \dots$ , and thus that the one time pad is not merely an  $\infty$ -alphabetic substitution cipher. It is also a limit of polyalphabetic substitution ciphers. It is torsion-free (i.e. lacking in nonzero elements of finite order [MA67, p. 81]). But it is not the only limit of polyalphabetic ciphers. Evidently for every nonzero rational number  $q$ , the group  $\mathbb{Q}/q\mathbb{Z}$  amounts (up to isomorphism) to  $\mathbb{Q}/\mathbb{Z}$ , the rationals [MA67, pp. 166-171] modulo 1. This group  $\mathbb{Q}/q\mathbb{Z}$  is the direct limit (i.e. inductive limit, colimit [MA71, pp. 67-68]. In this case, in a natural sense, the direct limit is the union [HA60, pp. 12-13] of the terms) of the sequence [HA60, p. 45]

$$\mathbb{Z}/\mathbb{Z}, \mathbb{Z}/2\mathbb{Z}, \mathbb{Z}/3\mathbb{Z}, \dots$$

This group is also the jumping-off point for the cipher key based on

$$m: \mathbb{Q} \rightarrow A/L$$

$$s: \mathbb{Q}/q\mathbb{Z} \rightarrow \text{SYM}(A/L)$$

$$y(e + \lambda) = s_{\lambda}(m(e + \lambda))$$

for every  $e \in q\mathbb{Z}$ , every  $\lambda \in \Delta$  (the set of coset leaders of  $\mathbb{Q}/q\mathbb{Z}$ ). This is also an  $\infty$ -alphabetic substitution cipher. But it is all torsion [MA67, pp. 344-348] (i.e. every element of  $\mathbb{Q}/q\mathbb{Z}$  is of finite order). It repeats its (infinitely) many permutations of its alphabet at intervals of  $q$ . This suggests yet another  $\infty$ -alphabetic substitution cipher key based on

$$m: \mathbb{R} \rightarrow A/L$$

$$s: \mathbb{R}/r\mathbb{Z} \rightarrow \text{SYM}(A/L)$$

$$y(e + \lambda) = s_{\lambda}(m(e + \lambda))$$

for every  $e \in r\mathbb{Z}$ , every  $\lambda \in \Delta$  (the set of coset leaders of  $\mathbb{R}/r\mathbb{Z}$ ), where  $r$  is any nonzero real number. The structure  $\mathbb{R}/r\mathbb{Z}$  has very little torsion. Only the rational multiples of  $r$  have finite order. And they form a set of Lebesgue measure [RO71, pp. 52-63]

zero. This cipher key repeats its alphabets at intervals of  $r$ . If these last two cipher keys (the one whose permutations of its alphabet are indexed by  $Q/qZ$  and the one whose permutations of its alphabet are indexed by  $R/rZ$ ) are equally easy to break, then torsion would seem to have little to do with the cryptanalysis of polyalphabetic substitution ciphers. If not, then torsion may play a role in such cryptanalysis.

## 6. TRANSPOSITION CIPHERS

A natural example of how cosets arise in transposition ciphers can be given in terms of a transposition cipher key which turns the message

HOMOMORPHISM

into the message

OMOHPRMMSIH.

One way to obtain this encryption is to reverse successive four letter strings

HOMO  $\rightarrow$  OMOH

MORP  $\rightarrow$  PROM

HISM  $\rightarrow$  MSIH

This is compatible with the definitions

$$m = \{(1,H), (2,0), \dots, (12,M)\} \cup \{(j,\text{blank}): j \notin \{1, 2, \dots, 12\}\}$$

and

$$t = \{(4w+1, 4w+4), (4w+2, 4w+3), (4w+3, 4w+2), (4w+4, 4w+1): w \in \mathbb{Z}\},$$

whence

$$m \circ t = \{(1,0), (2,M), \dots, (12,H)\} \cup \{(j,\text{blank}): j \notin \{1, 2, \dots, 12\}\}$$

Such transposition cipher keys are clearly of the form

$$y(e + \lambda) = m(f(e + \lambda)) = m(e + t(\lambda))$$

where  $f(e + \lambda) = e + t(\lambda)$  and

$$t \in \text{SYM}(\Lambda) = \text{SYM}(P/E)$$

is arbitrary. In the case at hand

$$P = Z$$

$$E = 4Z$$

$$P/E = Z/4Z = \Lambda = \{1, 2, 3, 4\}$$

$$t = \{(1,4), (2,3), (3,2), (4,1)\}.$$

## 7. MIXING AND "ASSOCIATIVITY"

The designer of a cryptosystem has no reason to be grateful for the associative law of function composition. Suppose, for example, that

$$q: A/L \rightarrow A/L$$

$$s: A/L \rightarrow A/L$$

are monalphabetic substitution cipher keys and that

$$t: P/E \rightarrow P/E$$

$$u: P/E \rightarrow P/E$$

are transposition cipher keys. Then we know from associativity that

$$q \circ (s \circ (m \circ (t \circ u))) = ((q \circ s) \circ m) \circ (t \circ u) = \dots = q \circ s \circ m \circ t \circ u.$$

Such combinations of keys exhibit what Hellman calls closure. Repeated operations do not enhance security. Whenever, on the other hand, one can contrive operations such that, for example,

$$q \circ ((s \circ (m \circ t)) \circ u) \neq ((q \circ s) \circ (m \circ t)) \circ u$$

or

$$s(m(\alpha + \beta)) \neq s(m(\alpha)) + s(m(\beta))$$

the possibility of greater cryptographic strength exists. The third thesis of this paper is that mixing (in the Shannon [SH49] sense) amounts to the failure of algebraic identities (such as commutative,

distributive or, especially, associative laws) which would make a cryptanalyst's job easier when dealing with cryptosystems which are compounded of a succession of confusion, diffusion, and replacement operations.

How can associativity fail? It cannot when transposition and monalphabetic substitution are the only operations used. But we also have both polyalphabetic substitution and replacement at our disposal.

Consider, first, a message

$$m \in (A/L) \uparrow P/E ,$$

a simple substitution

$$s \in (A/L) \uparrow A/L$$

and a replacement

$$r: (A/L) \uparrow P/E \rightarrow (A/L) \uparrow P/E .$$

The expression  $r(s \circ m)$  makes sense and, in fact,

$$s \circ m \in (A/L) \uparrow P/E ,$$

whence

$$r(s \circ m) \in (A/L) \uparrow (P/E)$$

makes perfectly good sense. But what can  $(r(s)) \circ m$  mean? After all

$$s \in (A/L) \uparrow (A/L)$$

but the domain of  $r$  is  $(A/L) \uparrow P/E$ . So there is usually no way to make sense of  $r(s)$ , much less  $(r(s)) \circ m$ . Consequently we conclude that an equality such as

$$r(s \circ m) = (r(s)) \circ m$$

is impossible (and, in fact, nonsensical) in all but very special and contrived circumstances. What about a comparison between  $r(m \circ t)$  and  $(r(m)) \circ t$ ? In this case both symbols make sense, and both

symbols belong to

$$(A/L) \uparrow P/E .$$

But are they equal? Not usually. For example let

$$P = A = Z ,$$

$$E = L = 2Z ,$$

whence

$$P/E = A/L = Z/2Z .$$

Let

$$m = \{(0,0), (1,1)\}$$

$$t = \{(0,1), (1,0)\}$$

$$r(\{(0,0), (1,1)\}) = \{(0,1), (1,1)\}$$

$$r(\{(0,1), (1,0)\}) = \{(0,0), (1,0)\}$$

Then

$$m \circ t = \{(0,1), (1,0)\}$$

$$r(m \circ t) = \{(0,0), (1,0)\}$$

$$r(m) = \{(0,1), (1,1)\}$$

$$(r(m)) \circ t = \{(0,1), (1,1)\}$$

whence

$$r(m \circ t) \neq (r(m)) \circ t .$$

An equation such as

$$r(m \circ t) = (r(m)) \circ t$$

is not, of course, a true associative law, since  $r(m)$  is the action of a function  $r$  on a "point"  $m$  of its domain, whereas  $m \circ t$  is the composite of the function  $m$  following the function  $t$ . The design of DES uses all three operations, confusion, diffusion and replacement. And it achieves mixing by exploiting such failures of "associativity" in its rounds.

"Associativity" can fail in other ways, too. We will content ourselves with merely mentioning one more example of failure of "associativity". The reader can easily verify the fact that polyalphabetic substitutions need not commute with transposition, even though monalphabetic substitution does, i.e. even though

$s \circ (m \circ t) = (s \circ m) \circ t$  when  $m$  is a message,  $s$  is a monalphabetic substitution, and  $t$  is a transposition.

## 8. NUMBER-THEORETIC CRYPTOSYSTEMS, MONALPHABETS AND CAESARS

The Pohlig-Hellman [PO78] conventional cryptosystem (PHC) and the Rivest-Shamir-Adleman [RI78] public key cryptosystem (RSA) are number-theoretic cryptosystems in the sense of [BL79]. PHC is the 1-prime case. Both the prime  $p$  and  $\phi(p) = \lambda(p) = p-1$  must be kept secret. RSA is the 2-prime case. The prime  $p$  and  $q$ , as well as the totient  $\phi(pq) = (p-1)(q-1)$  and the universal [LE56, Vol. 1, pp. 53-56] exponent  $\lambda(pq)$  must be kept secret. Both cryptosystems are monalphabetic substitutions with large Caesar subsystems.

To make this statement precise we sketch the definition of a general number-theoretic cryptosystem. So let

$$w = \prod p$$

be a (square-free) product of odd primes  $p$  belonging to a (secret) set  $P$  of primes. Since  $P$  is secret, the universal exponent  $\lambda(w)$  is secret.  $\lambda(w)$  is, by definition, the least common multiple of the members of the set  $\Lambda = \{p-1: p \in P\}$ .

For a given modulus  $w = \prod p$  there are  $\phi(\lambda(w))$  encode/decode pairs  $(c,d)$  of positive integers less than  $\lambda(w)$  such that

$$cd \equiv 1 \pmod{\lambda(w)}.$$

Encoding is the process

$$x \rightarrow x+c \bmod w$$

Decoding is

$$y \rightarrow y+d \bmod w.$$

Any key  $(c,d)$  in this cryptosystem thus amounts to an encryption process which is a permutation of  $\mathbb{Z}/w\mathbb{Z}$ . Number-theoretic cryptosystems are thus monalphabetic substitutions on the alphabet

$$A/L = \mathbb{Z}/w\mathbb{Z}$$

Since there are so few keys  $(c,d)$  corresponding to a given modulus  $w$  (i.e. to an alphabet  $A/L = Z/wZ$ ) one might suspect the existence of a Caesar cipher, perhaps on a subset of this alphabet. And one does exist. There are elements  $\gamma \in Z/wZ$  with (multiplicative) order  $\lambda(w)$ . Let  $\gamma$  be one of them and let  $\Gamma \subseteq Z/wZ$  be the set

$$\Gamma = \{\gamma, \gamma+2, \gamma+3, \dots, \gamma+\lambda(w) = 1\}$$

of powers of  $\gamma$ . Evidently  $\Gamma$  is a rather large subset of  $Z/wZ$ , since  $\lambda(w)/w$  is fairly close to 1 if  $w$  is the product of just a few primes  $p$  such that  $p-1$  does not have many small factors. The encryption process  $x \rightarrow x+c$  effects a permutation of the members of any such  $\Gamma$ , since  $c$  is relatively prime to  $\lambda(w)$ . Encryption on  $\Gamma$  is the mapping

$$\gamma+a \rightarrow (\gamma+a)+c = \gamma+(a*c) .$$

where the asterisk denotes multiplication modulo  $\lambda(w)$ . Thus the encryption process, restricted to  $\Gamma$ , is determined by the Caesar cipher mapping

$$a \rightarrow a*c .$$

But to exploit our knowledge of the existence of very large Caesar subsystems of an RSA public key cryptosystem [RI78] or a Pohlig-Hellman conventional cryptosystem [PO78] we appear to have to find some appropriate  $\gamma$ , as well as its corresponding set  $\Gamma$ , and be able to solve a corresponding discrete logarithm problem.

Let us take an RSA example. Let  $w = 35$ . The RSA is a monalphabet substitution cipher on the 35 member alphabet  $Z/35Z$ . In this case

$$\lambda(w) = \lambda(35) = 12$$

The  $\phi(\lambda(35)) = \phi(12) = 4$  keys amount to the 4 encrypt/decrypt exponent pairs

$$(c,d) = (1,1),$$

$$(c,d) = (5,5),$$

$$(c,d) = (7,7),$$

$$(c,d) = (11,11).$$

One such  $\Gamma$  is the set



$$\Gamma = \{2, 4, 8, 16, 32, 29, 23, 11, 22, 9, 18, 1\}$$

of all powers of  $\gamma = 2$ . Let us note what happens when the key (5,5) is used.

$$\begin{aligned} 2+5 &= (2+1)+5 = 2+(1*5) = 2+5 = 32 \\ 4+5 &= (2+2)+5 = 2+(2*5) = 2+10 = 9 \\ 8+5 &= (2+3)+5 = 2+(3*5) = 2+3 = 8 \\ 16+5 &= (2+4)+5 = 2+(4*5) = 2+8 = 11 \\ 32+5 &= (2+5)+5 = 2+(5*5) = 2+1 = 2 \\ 29+5 &= (2+6)+5 = 2+(6*5) = 2+6 = 29 \\ 23+5 &= (2+7)+5 = 2+(7*5) = 2+11 = 18 \\ 11+5 &= (2+8)+5 = 2+(8*5) = 2+4 = 16 \\ 22+5 &= (2+9)+5 = 2+(9*5) = 2+9 = 22 \\ 9+5 &= (2+10)+5 = 2+(10*5) = 2+2 = 4 \\ 18+5 &= (2+11)+5 = 2+(11*5) = 2+7 = 23 \\ 1+5 &= (2+12)+5 = 2+(12*5) = 2+0 = 1 \end{aligned}$$

So more than 1/3 of this RSA is a concealed version of the Caesar cipher

$$t \rightarrow 5t \text{ modulo } 12$$

acting on the set

$$\{1, 2, \dots, 10, 11, 0 = 12 = \lambda(35)\}$$

A different  $\gamma, \Gamma$  pair would be

$$\gamma = 3$$

$$\Gamma = \{3, 9, 27, 11, 33, 29, 17, 16, 13, 4, 12, 1\}.$$

A similar analysis can be provided for the Caesar cipher based on this  $\gamma, \Gamma$  pair.

It seems inappropriate to regard number-theoretic cryptosystems as weak because each of them has a huge subsystem  $\Gamma$  such that the cryptosystem operation on  $\Gamma$  is equivalent to a Caesar cipher key. We are left, rather, with a renewed respect for the much-maligned Caesar cipher because it can be transformed and inserted into a

number-theoretic cryptosystem in a natural way that, as of this writing, leaves it unbroken.

## 9. RSA and factorization

RSA is formulated with  $\mathbb{Z}$  and  $\mathbb{Z}/w\mathbb{Z}$  as the rings which draw most of our notice. We therefore pay a lot of attention to the problem of factoring  $w$  into the product of two primes  $p$  and  $q$  in the ring  $\mathbb{Z}$  of integers. But the group-theoretic approach is neutral as regards the ring in question. There are infinitely many factorizations of an RSA modulus  $w$ . Many, such as the trivial factorizations

$$w = 2 * (w/2)$$

or

$$w = \sqrt{3} * (w/\sqrt{3})$$

seem to hold out little promise to a cryptanalyst.

At Crypto '83 H. C. Williams suggested a somewhat more disciplined -- and perhaps more informative -- approach to factorization. It might be interesting to look at factorizations in the integral domain  $\mathbb{Q}[\theta]$  of an algebraic number [LE56, vol. 2, pp. 34-81] field  $\mathbb{Q}(\theta)$  (Here  $\mathbb{Q}$  is the field of rational numbers, and  $\theta$  is algebraic over  $\mathbb{Q}$ ). Such a factorization might [CR83] contain information sufficient to enable a cryptanalyst to calculate a large multiple  $k\phi(pq)$  of  $\phi(pq) = (p-1)(q-1)$ . This would be enough information to provide a (very large) decoding exponent.

The question of how to search for an appropriate  $\theta$  (perhaps of the form  $\theta = \sqrt{d}$  for  $d \in \mathbb{Z}$ ) and how to calculate a generalized Euler totient function in that  $\mathbb{Q}[\theta]$  is open. But it would seem that those who wish to use RSA might want to satisfy themselves that it does not yield to attacks of the  $\mathbb{Q}[\theta]$  type any more readily than to attacks made entirely within  $\mathbb{Z}$ .

## 10. DISCUSSION

The motivation behind this work was to extend cryptography to infinite structures by analogy with recent extensions of the notion of threshold scheme [BL83] to infinite structures. But it seems

necessary to justify the naturalness of the group-theoretic formulation as an abstraction arising out of consideration of many finite, as well as a few infinite, structures. So this paper dealt largely with finite examples, and very simple ones, to argue for the ubiquity of the

$$m: P/E \rightarrow A/L$$

structure of messages. In the analog case, to be considered elsewhere,  $P/E$  and  $A/L$  are more likely to be infinite groups such as  $R/OR = R$ , or  $R/2\pi Z$  (essentially the complex unit circle under multiplication).

We have given candidates for precise definitions of the rather intuitive notion of confusion (including substitution as a special case) and diffusion (including transposition). We have distinguished between a cryptosystem (a family of keys) and a key, i.e. a map which can be expressed in terms of confusion, diffusion, and replacement. This approach to key seems more mathematically natural than the old-fashioned viewpoint which regards a key as a number which, when entered into a crypto box, gives rise to the map this paper calls a key.

We have given a precise definition of Caesar cipher more general than the one commonly found [K081, pp. 69-72] in the literature, and have shown that Caesars are not as cryptographically trivial as the conventional wisdom dictates. The Caesar cipher illustrates well, in a confusion/substitution context, what we hope to exemplify elsewhere regarding diffusion/transposition, namely that the natural group operation on the domain (resp. codomain) is often the basis of the diffusion operator  $t$  (resp. confusion operator  $s$ ).

The maps encountered in the keys which make up most well-known cryptosystems are not morphisms. Indeed the less algebraic structure these maps exhibit, the more likely the cryptosystems employing them in keys are to be secure. This seems to suggest more reliance on nonabelian groups  $P$  and  $A$  both in the design of future cryptosystems and in the upgrading of existing cryptosystems. Perhaps, eventually, even more general structures (e.g. monoids, semigroups, etc.) might become useful in cryptosystem design.

More complicated finite cryptosystems, such as Polybius, Delastelle, Playfair [KA67] and the remarkably highly structured DES, require a deeper and more interesting elaboration of the topics introduced above. After that it will be natural to turn to infinite structures and to cryptanalysis. We will treat such topics elsewhere.

NSA Grant MDA-83-H-0002 supported this work.

## 11. REFERENCES

- AD83 L. M. Adleman, C. Pomerance and R. S. Rumely, On distinguishing prime numbers from composite numbers, *Annals of Mathematics*, vol. 117 (1983), pp. 173-206.
- BA64 R. G. Bartle, *The Elements of Real Analysis*, Wiley, New York (1964).
- BE82 H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*, Wiley-Interscience, New York (1982).
- BE83 G. R. Blakley and Laif Swanson, Infinite structures in information theory, in D. Chaum, R. L. Rivest and A. T. Sherman, *Advances in Cryptology, Proceedings of Crypto '82*, Plenum Press, New York (1983), pp. 39-50.
- BL79 Bob Blakley and G. R. Blakley, Security of number theoretic public key cryptosystems against random attack, Part I, *Cryptologia*, Vol. 2 (1978), pp. 305-321, Part II, Vol. 3 (1979), pp. 29-42, Part III, Vol. 3 (1979), pp. 105-118.
- CR83 J. T. Cross, The Euler  $\phi$  function in the Gaussian integers, *American Mathematical Monthly*, vol. 90 (1983), pp. 518-528.
- DE82 D. E. R. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, Massachusetts (1982).
- DI79 W. Diffie and M. E. Hellman, Privacy and authentication, An introduction to cryptography, *Proceedings of the IEEE*, vol. 67 (1979), pp. 397-427.
- GD58 C. Goffman, *Real Functions*, Rinehart, New York (1958).
- HA60 P. R. Halmos, *Naive Set Theory*, Van Nostrand, Princeton, New Jersey (1960).
- KA67 D. Kahn, *The Codebreakers*, MacMillan, New York (1967).
- LE56 W. J. LeVeque, *Topics in Number Theory*, Addison-Wesley, Reading, Massachusetts (1956).
- KO81 A. G. Konheim, *Cryptography: A Primer*, Wiley-Interscience, New York (1981).
- MA67 S. MacLane and G. Birkhoff, *Algebra*, Macmillan, New York (1967).
- MA71 S. MacLane, *Categories for the Working Mathematician*, Springer-Verlag, Berlin (1971).
- ME82 C. H. Meyer and S. M. Matyas, *Cryptography: A New Dimension in Computer Data Security*, Wiley-Interscience (1982)
- MO63 G. D. Mostow, J. H. Sampson and J.-P. Meyer, *Fundamental Structures of Algebra*, McGraw-Hill, New York (1963).

- PA66 H. Paley and P. M. Weichsel, A First Course in Abstract Algebra, Holt, Rinehart and Winston, New York (1966).
- PO78 S. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over  $GF(p)$  and its cryptographic significance, IEEE Transactions on Information Theory, Vol. IT-24 (1978), pp. 106-110.
- QU82 J.-J. Quisquater and C. Couvreur, Fast decipherment algorithm for RSA public-key cryptosystem, Electronics Letters, Vol. 18, No. 21, Oct. 14 (1982), pp. 905-907.
- RI78 R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Communication of the ACM, Vol. 21 (1978), pp. 120-126.
- RO71 H. L. Royden, Real Analysis, Macmillan, London (1971).
- SH49 C. E. Shannon, Communication theory of secrecy systems, Bell System Technical Journal, vol. 28, (1949), pp. 656-715.
- ST33 H. S. Stone, Discrete Mathematical Structures and their Applications, Science Research Associates, Chicago (1973).

# CRYPTANALYSIS OF ADFGVX ENCIPHERMENT SYSTEMS

Alan G. Konheim

Computer Science Department

University of California

Santa Barbara, California 93106 USA

## Extended Abstract

The ADFGVX cryptographic system, invented by Fritz Nebel, was introduced by Germany during World War I on March 5, 1918. The names ADFGX and ADFGVX for the successor system refer to the use of only five (and later six) letters A, D, F, G, (V,) X in the ciphertext alphabet. Kahn [KA] suggests that these letters were chosen because differences in Morse International symbols

A    • -

D    - • •

F    • • - •

G    - - •

V    • • • -

X    - • • -

aided the prevent misidentification due to transmission noise.

The ADFGVX system is historically important since it combined both letter substitution and fractionation (transposition). Although Allied cryptanalysts did not develop a general method for the solution of ADFGVX ciphertext, Georges Painvin of the French Military Cryptographic Bureau found solutions which significantly effected the military outcome in 1918. This paper proposes a new method for the cryptanalysis of ADFGVX-type systems.

Let  $A$  denote an alphabet of  $m = M^2$  "letters" which we henceforth identify with the set of integers  $Z_m = \{0, 1, \dots, m-1\}$ . The ADFGVX key ( $SUB, \pi$ ) has two components; the first, an  $M$  by  $M$  array  $SUB$  containing an arrangement of the letters of  $Z_m$ . For example, with  $m = 25$

$$SUB = \begin{vmatrix} C & R & Y & P & T \\ O & G & A & H & B \\ D & E & F & I & K \\ L & M & N & Q & S \\ U & V & W & X & Z \end{vmatrix}$$

The second is a transposition

$$\pi = (\pi(0), \pi(1), \dots, \pi(N-1))$$

on  $N$  places.

The steps in an ADFGVX encipherment are as follows:

**ADFGVX(1):** Plaintext of  $n$   $m$ -letters

$$\overline{x} = (x_0, x_1, \dots, x_{n-1}) \quad x_i \in \mathbf{Z}_m$$

is expanded into a  $2n$ -gram of  $M$ -letters

$$\overline{z} = (z_0, z_1, \dots, z_{2n-1}) \quad (z_{2i}, z_{2i+1}) = (x_{i,0}, x_{i,1}) \quad x_{i,j} \in \mathbf{Z}_M$$

The  $\{x_i\}$  are determined by the substitution *SUB*

$$x_i \rightarrow (x_{i,0}, x_{i,1}) \quad x_{i,0}, x_{i,1} \in \mathbf{Z}_M$$

where  $x_{i,0}$  and  $x_{i,1}$  are the row and columns coordinates of  $x_i$  in *SUB*.

**ADFGVX(2):** The "expanded" plaintext  $\overline{z}$  is arranged in a (possibly) "ragged"  $z$ -array containing  $r$  rows of  $N$  columns and a (possible)  $(r+1)$ st "short" row of  $s < N$  columns;

$$2n = rN + s \quad (0 \leq s < N)$$

$$z : \begin{vmatrix} z_0 & z_1 & \dots & \dots & z_{N-1} \\ z_N & z_{N+1} & \dots & \dots & z_{2N-1} \\ \cdot & \cdot & \dots & \dots & \cdot \\ \cdot & \cdot & \dots & \dots & \cdot \\ \cdot & \cdot & \dots & \dots & \cdot \\ z_{(r-1)N} & z_{(r-1)N+1} & \dots & \dots & z_{rN-1} \\ z_{rN} & \dots & \dots & \dots & z_{rN+s} \end{vmatrix}$$

**ADFGVX(3):** The ciphertext  $\overline{y} = (y_0, y_1, \dots, y_{2n-1})$  is the concatenation of the columns of the  $z$ -array in the order defined by  $\pi$ .

We assume the length  $N$  of the transposition  $\pi$  is known, although the method will suggest a procedure to test a value  $N$  as a presumptive transposition length. The ciphertext

$$\overline{y} = (y_0, y_1, \dots, y_{2n-1}) \quad 2n = rN + s$$

is the concatenation of segments  $\{\overline{y}^{(i)}\}$  of  $\overline{y}$  which correspond to the entries in a single column of the  $z$ -array. We call  $\overline{y}^{(i)}$  a *column vector*. The cryptanalysis will follow these steps:

**Step 1:** Determine which column vectors  $\{\overline{y}^{(i)}\}$  are adjacent in the  $z$ -array.

**Step 2:** Determine the relative order of the pair  $\overline{y}^{(\alpha_i)} \overline{y}^{(\beta_i)}$  of adjacent column vectors

$$\overline{y}^{(\alpha_i)} \overline{y}^{(\beta_i)} \quad \text{or} \quad \overline{y}^{(\beta_i)} \overline{y}^{(\alpha_i)}$$

**Step 3:** Recover the substitution *SUB*.

**Step 4:** Recover the transposition  $\pi$ .

To carry out *Step 1*, we detect the "dependence" between the marginal "letter counts"  $N_i^{(i)}$ ,  $N_i^{(j)}$  and  $N_{s,t}^{(i,j)}$  for a pair of column vector  $\bar{y}^{(i)}$ ,  $\bar{y}^{(j)}$  where

$$N_i^{(i)} = \sum_{t=0}^{M-1} N_{s,t}^{(i,j)} \quad N_i^{(j)} = \sum_{s=0}^{M-1} N_{s,t}^{(i,j)}$$

and  $N_{s,t}^{(i,j)}$  is equal to the number of solutions  $k = 0, 1, \dots$  of

$$y_{ir+k} = s \quad y_{jr+k} = t \quad 0 \leq s, t < M$$

Dependence will be detected by a variant of the  $\chi^2$ -test.

Having identified and ordered (*Step 2*) adjacent column vectors  $\bar{y}^{(\alpha_i)}$ ,  $\bar{y}^{(\beta_i)}$ , the sum

$$N_{s,t} = \sum_i N_{s,t}^{(\alpha_i, \beta_i)}$$

is the count of m-letters  $(s, t) \in \mathbf{Z}_M \times \mathbf{Z}_M = \mathbf{Z}_m$  characteristic of a monalphabetic substitution. *SUB* may then be recovered by standard techniques. Having removed the effect of the substitution, the arrangement of the column vector pairs  $\{\bar{y}^{(\alpha_i)}, \bar{y}^{(\beta_i)}\}$  to reconstitute the z-array requires the solution of a pure transposition system.

The analysis requires an examination of several cases:

- Case 1:*  $N \equiv 0 \pmod{2}$   $s = 0$
- Case 2:*  $N \equiv 0 \pmod{2}$   $0 < s < N$
- Case 3:*  $N \equiv 1 \pmod{2}$   $s = 0$
- Case 4:*  $N \equiv 1 \pmod{2}$   $0 < s < N$

Details and proofs will appear in a paper submitted to the *IEEE Transactions on Information Theory*.



# BREAKING ITERATED KNAPSACKS\*

Ernest F. Brickell

Sandia National Laboratories  
Albuquerque, New Mexico 87185

## ABSTRACT

This paper presents an outline of an attack that we have used successfully to break iterated knapsacks. Although we do not provide a proof that the attack almost always works, we do provide some heuristic arguments. We also give a detailed description of the examples we have broken.

## INTRODUCTION

R. Merkle and M. Hellman [10] devised the first knapsack based cryptosystem. In this paper we will deal only with the Merkle-Hellman knapsack, although similar techniques will work on Graham-Shamir iterated knapsacks also. We will say that a set of positive integers  $a_1, \dots, a_n$  is an ordered  $Y$ -times iterated knapsack if there exists a superincreasing sequence  $s_1, \dots, s_n$  (i.e.,  $s_i > \sum_{j < i} s_j$ ), and integers  $w_j^*, M_j, a_{j,i}$  for  $1 \leq j \leq Y$  and  $1 \leq i \leq n$  such that

$$a_{0,i} = s_i \quad \text{for } 1 \leq i \leq n, \quad (1.1)$$

$$a_{j,i} = a_{j-1,i} w_j^* \bmod M_j \quad \text{for } 1 \leq j \leq Y \text{ and } 1 \leq i \leq n, \quad (1.2)$$

$$M_j > \sum_{i=1}^n a_{j-1,i} \quad \text{for } 1 \leq j \leq Y, \quad (1.3)$$

$$a_i = a_{Y,i} \quad \text{for } 1 \leq i \leq n. \quad (1.4)$$

---

\* This work performed at Sandia National Laboratories supported by the U. S. Department of Energy under Contract Number DE-AC04-76DP00789.

We use  $a = b \bmod M$  to mean that  $a$  is the least nonnegative residue. Let  $W_j = W_j^{*-1} \bmod M_j$  for  $1 < j < Y$ . We will define integers  $k_{j,i}$  for  $1 < j < Y$  and  $1 < i < n$  to be those integers satisfying

$$a_{j,i} W_j - k_{j,i} M_j = a_{j-1,i} \quad \text{for } 1 < j < Y \quad \text{and } 1 < i < n. \quad (1.5)$$

We will say that a set of positive integers  $a_1, \dots, a_n$  is an unordered  $Y$ -times iterated knapsack if there is some permutation of the  $a_1, \dots, a_n$  that is an ordered  $Y$ -times iterated knapsack.

To cryptanalyze this system, one must solve the knapsack (or subset sum) problem for the integers  $a_1, \dots, a_n$  and any subset sum  $s$ . That is given  $s$ , one must find a 0-1 vector  $(\alpha_1, \dots, \alpha_n)$  such that

$$s = \sum_{i=1}^n \alpha_i a_i$$

if such a 0-1 vector exists.

In fact it is sufficient to be able to solve the knapsack problem for a 0-1 vector  $(\alpha_1, \dots, \alpha_n)$  which has  $< \frac{1}{2} n$  ones, because one can consider the subset sum  $s$  and the subset sum  $\sum_{i=1}^n a_i - s$ .

In this paper we will describe an algorithm for breaking  $Y$ -times iterated knapsacks in polynomial time. We have successfully demonstrated this algorithm on examples with  $n = 100$  and  $Y = 5, 10$ , and  $20$ .

The first attack on knapsack based cryptosystems was found by Adi Shamir [13]. He discovered an algorithm for cryptanalyzing the single iteration Merkle-Hellman knapsack in polynomial time. Len Adleman [1] found a method for breaking the single iteration Graham-Shamir knapsack in polynomial time. His attack used the Lenstra, Lenstra, Lovász ( $L^3$ ) lattice basis reduction algorithm [9], which could also be used to greatly speed up the attack on the Merkle-Hellman knapsack. Len Adleman [1] and Jeff Lagarias [7] have both developed attacks for the doubly iterated Merkle-Hellman knapsack. Adleman [1] also proposed an attack on multiply iterated knapsacks, but Brickell, Lagarias, and Odlyzko [3] showed that there were some problems with it. However the lattice

that we use in our new attack is the same as the first lattice that Adleman used in his attack.

By using a different approach, E. Brickell [2] and J. Lagarias and A. Odlyzko [8] have developed algorithms which will cryptanalyze MH or GS cryptosystems in polynomial time if the information rate is low enough. The information rate of a knapsack based cryptosystem is roughly  $n/\log_2(\max a_{Y,i})$ . It is not known exactly how low the information rate must be for these algorithms to work, but the information rate must at least be less than .645 before these methods can possibly be successful. Since each iteration lowers the information rate, these methods will break a knapsack cryptosystem if it has been iterated too many times.

#### THE USE OF LATTICE REDUCTION

The  $L^3$  lattice basis reduction algorithm [9] is used in all of the attacks on knapsack based cryptosystems. A set of points,  $L$ , in  $\mathbb{R}^n$  is a lattice if there exists a set of independent vectors  $v_1, \dots, v_m$  such that

$$L = \{z_1 v_1 + \dots + z_n v_n : z_i \in \mathbb{Z} \text{ for } 1 \leq i \leq m\}.$$

Such a set of vectors,  $v_1, \dots, v_n$  is called a basis for the lattice. The  $L^3$  algorithm finds a reduced (or short) basis for the lattice. We will not give a precise definition of a reduced basis. We will only note that in a reduced basis, all of the basis vectors are relatively short in the Euclidean norm.

Let  $a_1, \dots, a_n$  be an unordered  $Y$ -times iterated knapsack. Let  $m$  be an integer  $< n$ . Later we will put conditions on how small  $m$  can be. Let  $L$  be the  $m$ -dimensional lattice generated by the following vectors.

$$\begin{aligned} b_1 &= (a_2, a_3, a_4, \dots, a_m, n^{-1}) \\ b_2 &= (a_1, 0, 0, \dots, 0, 0) \\ b_3 &= (0, a_1, 0, \dots, 0, 0) \\ &\vdots \\ b_m &= (0, 0, 0, \dots, a_1, 0) \end{aligned}$$

The first step in the algorithm is to find a reduced basis for  $L$  by using the  $L^3$  algorithm. We now must go into a rather detailed discussion to describe the vectors we expect to see in the reduced basis.

Let  $D$  be the smallest integer such that  $2^D > \max\{a_1, \dots, a_n\}$ . Then  $M_Y, W_Y, a_1, \dots, a_n$  should be  $O(2^D)$ . Since  $M_j > \sum_{i=1}^n a_{j-1,i}$ , we will assume that  $M_{Y-k}, W_{Y-k}, a_{Y-k,i}$  are  $O(2^{D-k} \log n)$ .

The norm of the vector

$$w = -a_1 b_1 + \sum_{i=2}^m a_i b_i = (0, 0, \dots, 0, \frac{-a_1}{n})$$

is  $O(2^{D-\log n})$ . If we take  $m > \frac{D}{\log n - 1}$ , then  $w$  would probably be the shortest vector in  $L$  if the integers  $a_1, \dots, a_n$  were chosen from the uniform distribution on  $(0, 2^D)$ . However, because  $a_1, \dots, a_n$  are an unordered  $Y$ -times iterated knapsack, there are other vectors in  $L$  that are about the same length as  $w$ . For this discussion we will refer to any vector in  $L$  with norm  $< O(2^{D-\log n})$  as a short vector.

Adleman found that there was a short vector that was a result of the last iteration. From (1.5)

$$a_i W_Y - k_{Y,i} M_Y = a_{Y-1,i} \quad .$$

Divide by  $M_Y a_i$

$$\frac{W_Y}{M_Y} - \frac{k_{Y,i}}{a_i} = \frac{1}{M_Y} \left( \frac{a_{Y-1,i}}{a_i} \right) .$$

Subtract equation  $i$  from equation 1

$$\frac{k_{Y,i}}{a_i} - \frac{k_{Y,1}}{a_1} = \frac{1}{M_Y} \left( \frac{a_{Y-1,1}}{a_1} - \frac{a_{Y-1,i}}{a_i} \right) .$$

Multiply by  $a_1 a_i$

$$k_{Y,i} a_1 - k_{Y,1} a_i = \frac{1}{M_Y} (a_i a_{Y-1,1} - a_1 a_{Y-1,i}) \quad . \quad (2.1)$$

$$k_{Y,i}a_1 - k_{Y,1}a_i = O(2^{D-\log n}) .$$

In the vector

$$x = -k_{Y,1}b_1 + \sum_{i=2}^m k_{Y,i}b_i$$

each coordinate is  $O(2^{D-\log n})$ . So

$$|x| = O(2^{D-\log n}) .$$

Lagarias [7] found a description for short vectors in  $L$  that are the result of many iterations. Let  $t$  be an integer with  $1 < t < Y$ . By applying equation (1.5) repeatedly we get

$$a_i W_Y \dots W_t - k_{Y,k} M_Y W_{Y-1} \dots W_t - k_{Y-1,i} M_{Y-1} W_{Y-2} \dots W_t - \dots - k_{t,i} M_t = a_{t-1,i} .$$

We divide by  $a_i M_Y W_{Y-1} W_{Y-2} \dots W_t$  to get

$$\frac{W_Y}{M_Y} - \frac{1}{a_i} \left[ k_{Y,i} + k_{Y-1,i} \frac{M_{Y-1}}{M_Y W_{Y-1}} + k_{Y-2,i} \frac{M_{Y-2}}{M_Y W_{Y-1} W_{Y-2}} + \dots + k_{t,i} \frac{M_t}{M_Y W_{Y-1} W_{Y-2} \dots W_t} \right] = \frac{a_{t-1,i}}{a_i M_Y W_{Y-1} W_{Y-2} \dots W_t} .$$

Using the theory of simultaneous Diophantine approximation, there exists many sets of integers  $(r_1, \dots, r_Y)$  such that for  $1 < j < Y$

$$r_j = 0$$

or (2.2)

$$\frac{r_j}{r_Y} \sim \frac{M_j}{M_Y W_{Y-1} \dots W_j} .$$

Each of these vectors will give rise to a short vector in the lattice. For if we let

$$h_i = \sum_{j=1}^Y k_{j,i} r_j - r_0 a_i \quad \text{for } 1 < i < m , \quad (2.3)$$

then the vector

$$-hb_1 + \sum_{i=2}^m h_i b_i \quad (2.4)$$

will be a short vector in the lattice. We will call such a vector a desirable vector.

We would like to have  $Y$  desirable vectors in the reduced basis. Let  $\pi$  be a permutation on the first  $n$  integers such that  $a_{\pi(1)}, \dots, a_{\pi(n)}$  is an ordered  $Y$ -times iterated knapsack. We can pick any  $m$  of the weights  $a_1, \dots, a_n$ . If we pick these weights so that we do not pick  $a_{\pi(n)}, a_{\pi(n-1)}, a_{\pi(n-2)}$ , then we expect to get  $Y$  desirable vectors in the reduced basis.

The reason that we do not want  $a_{\pi(n)}, a_{\pi(n-1)}$ , or  $a_{\pi(n-2)}$  is because the  $Y$  short vectors exist because  $\frac{a_{j-1,i}}{M_j}$  is small for  $j=1, \dots, Y$  and all  $i$  in the weights that we pick. But

$$\frac{a_{0,\pi(n)}}{M_1} \approx \frac{1}{2} \quad \text{and} \quad \frac{a_{0,\pi(n-2)}}{M_1} \approx \frac{1}{8}$$

and these ratios are not small enough. In the examples that we tried, if  $a_{\pi(n)}, a_{\pi(n-1)}$ , or  $a_{\pi(n-2)}$  were in the chosen set, then we only had  $Y-1$  short vectors. But if these three weights were not in the chosen set and  $a_{\pi(n-3)}$  was, then we still had  $Y$  short vectors.

This condition on the way we must choose an  $m$ -set will not seriously affect the running time. The probability of picking a good  $m$ -set is

$$p = \frac{\binom{n-3}{m}}{\binom{n}{m}} \approx \left(\frac{n-m}{n}\right)^3.$$

Thus we expect to make about  $\left(\frac{n}{n-m}\right)^3$  choices to get a good  $m$ -set.

The information that we obtain from these short vectors are the coefficients  $h_i$  which we can recover from (2.4). We only know  $h_i$  for  $1 < i < m$ , and these  $h_i$  satisfy

$$\left| \frac{h_1}{a_1} - \frac{h_i}{a_i} \right| < \frac{1}{n} \left( \frac{1}{M_Y} \right) \quad \text{for} \quad 1 < i < m .$$

(We haven't proven this, but it has been true in all of our examples.)

We can define  $h_i$  for  $i > m$  by

$$h_i = \left[ \frac{h_1}{a_1} a_i \right] \quad \text{for} \quad m < i < n ,$$

where  $[x]$  is the closest integer to  $x$ . Because the  $h_i$ , for  $1 < i < m$ , have the special form of (2.3) and the  $r_j$  have the special form of (2.2),

$$\left| \frac{h_1}{a_1} - \frac{h_i}{a_i} \right| < \frac{1}{n} \left( \frac{1}{M_Y} \right) \quad \text{for} \quad 1 < i < n .$$

To get the full impact of the power of these vectors  $(h_1, \dots, h_n)$ , let  $s$  be a subset sum such that

$$s = \sum_{i=1}^n \alpha_i a_i$$

where  $(\alpha_1, \dots, \alpha_n)$  is a 0-1 vector with  $< \frac{1}{2} n$  ones. If we then define

$$t = \left[ \frac{h_1}{a_1} s \right] , \quad (2.5)$$

then

$$t = \left[ \frac{h_1}{a_1} \sum_{i=1}^n \alpha_i a_i \right] = \left[ \sum_{i=1}^n \alpha_i \frac{h_1}{a_1} a_i \right]$$

but

$$\frac{h_1}{a_1} a_i = h_i + \varepsilon_i$$

where

$$|\varepsilon_i| < \frac{1}{n} .$$

So

$$t = \sum_{i=1}^n \alpha_i h_i + \sum_{i=1}^n \alpha_i \varepsilon_i .$$

But

$$\sum_{i=1}^n \alpha_i \epsilon_i < \frac{1}{2}.$$

So

$$t = \sum_{i=1}^n \alpha_i h_i.$$

Thus we can find  $\sum_{i=1}^n \alpha_i h_i$  without knowing what the  $\alpha_i$  are.

#### THE USE OF DESIRABLE VECTORS

In this section we will show how the short vectors in the reduced basis can be used to recover a superincreasing sequence of length  $n-Y-\epsilon$ . We will comment about  $\epsilon$  later, but for now, assume that  $\epsilon$  is a small integer. In this section we will assume that  $a_1, \dots, a_n$  is an ordered  $Y$ -times iterated knapsack. Suppose that we have reduced the lattice and we have  $Y$  desirable vectors of the form

$$w_\ell = -h_{\ell,1}b_1 + \sum_{i=2}^n h_{\ell,i}b_i \quad 1 \leq \ell \leq Y$$

such that

$$h_{\ell,i} = \sum_{j=1}^Y r_{\ell,j} k_{j,i} - r_{\ell,0} a_i \quad \begin{matrix} 1 \leq \ell \leq Y \\ 1 \leq i \leq n \end{matrix}$$

and  $r_{\ell,j} = 0$  or  $\frac{r_{\ell,j}}{r_{\ell,Y}}$  is a good approximation to  $\frac{M_j}{M_Y W_{Y-1} \dots W_j}$ .

For  $1 \leq i \leq n$ , let

$$H_i = \begin{pmatrix} a_1 & \dots & a_Y & a_i \\ h_{1,1} & \dots & h_{1,Y} & h_{1,i} \\ \vdots & & & \\ h_{Y,1} & \dots & h_{Y,Y} & h_{Y,i} \end{pmatrix}$$

$$K_i = \begin{pmatrix} a_1 & \dots & a_Y & a_i \\ k_{1,1} & \dots & k_{1,Y} & k_{1,i} \\ \vdots & & & \\ k_{Y,1} & \dots & k_{Y,Y} & k_{Y,i} \end{pmatrix}.$$



Let

$$R = \begin{pmatrix} 1 & 0 & \dots & 0 \\ -r_{1,0} & r_{1,1} & \dots & r_{1,Y} \\ \vdots & \vdots & \ddots & \vdots \\ -r_{Y,0} & r_{Y,1} & \dots & r_{Y,Y} \end{pmatrix}.$$

Then

$$RK_i = H_i.$$

The following theorem gives us a way of computing the determinant of  $K_i$ .

Theorem:

Let  $a_1, \dots, a_n$  be an unordered  $Y$ -times iterated knapsack. Let  $1 < j < Y$ . Then

$$M_1 \dots M_Y \begin{vmatrix} a_{j,1} & \dots & a_{j,j+1} \\ k_{1,1} & \dots & k_{1,j+1} \\ \vdots & & \vdots \\ k_{j,1} & \dots & k_{j,j+1} \end{vmatrix} = \begin{vmatrix} a_{0,1} & \dots & a_{0,j+1} \\ a_{1,1} & \dots & a_{1,j+1} \\ \vdots & & \vdots \\ a_{j,1} & \dots & a_{j,j+1} \end{vmatrix}$$

Proof:

The case  $j=1$  follows immediately from (2.1). We will complete the proof by induction. Since any permutation of  $a_1, \dots, a_n$  is an unordered  $Y$ -times iterated knapsack, we can apply the inductive hypothesis to any permutation of  $a_1, \dots, a_n$ . So we get equality in the following statement by expanding about the last row and seeing that the cofactors are all equal.

$$\begin{vmatrix} a_{0,1} & \dots & a_{0,j+1} \\ \vdots & & \vdots \\ a_{j-1,1} & \dots & a_{j-1,j+1} \\ a_{j,1} & \dots & a_{j,j+1} \end{vmatrix} = M_1 \dots M_{j-1} \begin{vmatrix} a_{j-1,1} & \dots & a_{j-1,j+1} \\ k_{1,1} & \dots & k_{1,j+1} \\ \vdots & & \vdots \\ k_{j-1,1} & \dots & k_{j-1,j+1} \\ a_{j,1} & \dots & a_{j,j+1} \end{vmatrix}$$

$$= -M_1 \dots M_{j-1} \begin{vmatrix} a_{j,1} & \dots & a_{j,j+1} \\ k_{1,1} & \dots & k_{1,j+1} \\ \vdots & & \vdots \\ k_{j-1,1} & \dots & k_{j-1,j+1} \\ a_{j-1,1} & \dots & a_{j-1,j+1} \end{vmatrix}$$

$$= -M_1 \dots M_{j-1} \begin{vmatrix} a_{j,1} & \dots & a_{j,j+1} \\ k_{1,1} & \dots & k_{1,j+1} \\ \vdots & & \vdots \\ k_{j-1,1} & \dots & k_{j-1,j+1} \\ a_{j,1}W_j^{-k_{j,1}M_j} & \dots & a_{j,j+1}W_j^{-k_{j,j+1}M_j} \end{vmatrix}$$

$$= M_1 \dots M_j \begin{vmatrix} a_{j,1} & \dots & a_{j,j+1} \\ k_{1,1} & \dots & k_{1,j+1} \\ \vdots & & \vdots \\ k_{j-1,1} & \dots & k_{j-1,j+1} \\ k_{j,1} & \dots & k_{j,j+1} \end{vmatrix}.$$

Once again, let us take  $a_1, \dots, a_n$  to be an ordered  $Y$ -times iterated knapsack. For  $1 < i < n$ , let

$$A_i = \begin{pmatrix} s_1 & \dots & s_Y & s_i \\ a_{1,1} & \dots & a_{1,Y} & a_{1,i} \\ \vdots & & \vdots & \vdots \\ a_{Y,1} & \dots & a_{Y,Y} & a_{Y,i} \end{pmatrix}$$

and

$$x_i = |\det(A_i)|. \quad (3.1)$$

For  $i < Y$ ,  $x_i = 0$  since the last column in  $A_i$  is identical to another column. However in all of the examples we have run, the sequence  $x_i$  has been superincreasing for  $i > Y + \epsilon$ , where  $\epsilon < 2$ . Furthermore, since the cryptanalyst can compute  $\det(H_i)$ , he can compute the sequence  $cx_i$  where  $c = \det(R)/M_1 \dots M_Y$ .

Let us give a heuristic argument for why we might expect  $x_i$  to be superincreasing for  $i > Y + \epsilon$ . Expand  $A_i$  about the first row to get

$$\det(A_i) = s_1 A_{1,1}^i + \dots + s_Y A_{1,Y}^i + s_i A_{1,Y+1}^i \quad (3.2)$$

where  $A_{1,j}^i$  is the  $1,j$  cofactor of  $A_i$ . Let  $B_{1,j}^i$  be the submatrix of  $A_i$  formed by deleting the first row and  $j^{\text{th}}$  column of  $A_i$ . So

$$|A_{1,j}^i| = |\det B_{1,j}^i|.$$

Each entry in the  $1^{\text{th}}$  row of  $B_{1,j}^i$  is less than  $M_1$ . We expect that the distribution of the values of the  $A_{1,j}^i$  to be independent of  $i$  and  $j$  since the distribution of the entries in  $B_{1,j}^i$  is independent of  $i$  and  $j$ . But the sequence  $s_i$  is superincreasing, so there should be an  $\epsilon$  such that for  $i > Y + \epsilon$ , the expression for  $\det(A_i)$  in (3.2) is dominated by the last term.

#### RECOVERING THE ORDER

In the previous section, we showed that if a cryptanalyst had an ordered  $Y$ -times iterated knapsack, then he could recover a superincreasing sequence. However, a cryptanalyst will usually be faced with an unordered  $Y$ -times iterated knapsack. So in this section we will show how we can apply techniques similar to those of the last section to find the order in an unordered  $Y$ -times iterated knapsack.

Let  $a_1, \dots, a_n$  be an unordered  $Y$ -times iterated knapsack. Let  $\pi$  be a permutation such that  $a_{\pi(1)}, \dots, a_{\pi(n)}$  is an ordered  $Y$ -times iterated knapsack, i.e.,  $a_{\pi(n)}$  corresponds to  $s_n$ , the largest element in the superincreasing sequence. We will present a method for finding  $\pi(i)$  for  $i > Y + \epsilon$ , for  $\epsilon$  as described in the previous section.

Suppose we have already found  $\pi(n), \dots, \pi(n-j)$  for some  $j$ ,  $-1 < j < n-Y-\epsilon-2$ . (If  $j = -1$ , we haven't found anything yet.) Let  $I_j = \{1, \dots, n\} \setminus \{\pi(n), \dots, \pi(n-j)\}$ . We will pick many  $(Y+1)$ -subsets of  $I_j$ . For each subset  $j_0, \dots, j_Y \subseteq I_j$  that we pick, we will form a determinant.

$$x_{j_0, \dots, j_Y} = \begin{vmatrix} a_{j_0} & \dots & a_{j_Y} \\ h_{1,j_0} & \dots & h_{1,j_Y} \\ \vdots & & \vdots \\ h_{Y,j_0} & \dots & h_{Y,j_Y} \end{vmatrix} = c \begin{vmatrix} s_{j_0} & \dots & s_{j_Y} \\ a_{1,j_0} & \dots & a_{1,j_Y} \\ \vdots & & \vdots \\ a_{Y,j_0} & \dots & a_{Y,j_Y} \end{vmatrix}. \quad (4.1)$$

The reason we use these determinants is essentially the same as the reason we gave in the last section for expecting the  $x_i$  to be superincreasing. When we expand the right hand matrix in (4.1) about the first row, we expect the distribution on the values of the cofactors to be independent of the choice of  $j_0, \dots, j_Y$ . So the value of  $|x_{j_0}, \dots, j_Y|$  should be dominated by the largest  $s_{j_i}$  in the first row.

After picking many  $(Y+1)$ -subsets of  $I_j$ , we will keep the subset  $j_0, \dots, j_Y$  that gives the smallest determinant  $X_{j_0, \dots, j_Y}$ . We expect that the set  $j_0, \dots, j_Y$  does not contain any of  $\pi(n-j-1), \dots, \pi(n-j-l)$  for some value of  $l$  which will depend on  $n, j$ , and  $Y$ . Thus we form the sequence

$$z_i = \begin{vmatrix} a_{j_1} & \dots & a_{j_Y} & a_i \\ h_{1,j_1} & \dots & h_{1,j_Y} & h_{1,i} \\ & & \vdots & \\ h_{Y,j_1} & \dots & h_{Y,j_Y} & h_{Y,i} \end{vmatrix}$$

for  $1 \leq i \leq n$ . For some small  $\lambda$ , the sequence  $z_{\pi(n-j-l+\lambda)}, \dots, z_{\pi(n)}$  should be superincreasing, and the other values of  $z_i$  should be less than  $z_{\pi(n-j-l+\lambda)}$ . So we should be able to identify  $\pi(n-j-l+\lambda), \dots, \pi(n-j-1)$  and also check to see if our choice of  $\pi(n-j), \dots, \pi(n)$  was correct. We will set  $j \leftarrow j+l-\lambda$  and continue with this iterative process. We continue until  $j$  does not change.

This method for recovering the order worked much better in practice than we expected. In our examples, we always were able to iterate until  $j$  was less than  $Y+5$ . In other words we were able to find an  $\epsilon$ -superincreasing sequence with  $\epsilon < 5$ .

#### SOLVING FOR THE $\alpha_i$

All of the previous analysis has used only the weights  $a_1, \dots, a_n$ . In this section we show how to finish the cryptanalysis. That is, given  $s$  find a 0-1 vector  $\alpha_i$  that has less than  $\frac{1}{2}n$  ones such that

$$\sum_{i=1}^n \alpha_i a_i = s,$$

if such a vector exists. We will assume that we have integers  $h_{j,i}$  for  $1 < j < Y$  and  $1 < i < n$  such that the sequence

$$x_i = \begin{bmatrix} a_1 & \dots & a_Y & a_i \\ h_{1,1} & \dots & h_{1,Y} & h_{1,i} \\ \vdots & & & \\ h_{Y,1} & \dots & h_{Y,Y} & h_{Y,i} \end{bmatrix}$$

is superincreasing for  $i > Y + \epsilon$  and we can find  $t_j$  (from (2.5)) such that

$$t_j = \sum_{i=1}^n \alpha_i h_{j,i}.$$

We will find the  $\alpha_i$  in three steps.

Step 1:  $i > Y + \epsilon$

Let

$$t = \begin{bmatrix} a_1 & \dots & a_Y & s \\ h_{1,1} & \dots & h_{1,Y} & t_1 \\ \vdots & & & \\ h_{Y,1} & \dots & h_{Y,Y} & t_Y \end{bmatrix}$$

then

$$t = \sum_{i=1}^n \alpha_i x_i.$$

Since  $x_i$  is superincreasing for  $i > Y + \epsilon$ , we can easily find  $\alpha_i$  for  $i > Y + \epsilon$ .

Step 2:  $Y < i < Y + \epsilon$

To find  $\alpha_i$ , we must solve a knapsack problem with about  $\epsilon$  weights. We know  $\alpha_i$  for  $i > Y + \epsilon$ , and  $x_i = 0$  for  $i < Y$ , so we can find

$$t' = \sum_{i=1}^n \alpha_i x_i - \sum_{i=Y+\epsilon+1}^n \alpha_i x_i = \sum_{i=Y+1}^{Y+\epsilon} \alpha_i x_i$$

So we hope that  $\varepsilon$  is small. We conjecture that  $\varepsilon$  is bounded by  $O(\log n)$ . In our examples, we always had  $\varepsilon < 5$ .

Step 3:  $i < Y$

Since we now know  $a_i$  for  $i > Y$ , let

$$s' = s - \sum_{i=Y+1}^n a_i a_i$$

$$t'_j = t_j - \sum_{i=Y+1}^n a_i h_{j,i} \quad 1 < j < Y.$$

Let

$$s = (s', t'_1, \dots, t'_Y)$$

and

$$Z = \begin{pmatrix} a_1 & \dots & a_Y & a_{Y+1} \\ h_{1,1} & \dots & h_{1,Y} & h_{1,Y+1} \\ \vdots & & & \\ h_{Y,1} & \dots & h_{Y,Y} & h_{Y,Y+1} \end{pmatrix}.$$

The problem we are left with is to find a 0-1 vector  $\alpha = (a_1, \dots, a_{Y+1})$ , if one exists, such that

$$Z\alpha = s.$$

If  $\det(Z) \neq 0$ , then this problem is easily solved. In all of our examples, we have found that  $\det(Z) \neq 0$ . We can speed up this operation by computing  $Z^{-1} \bmod p$  where  $p$  is the smallest prime such that  $\det(Z) \neq 0 \bmod p$ . Then we are computing using integers less than  $p$  instead of integers of size  $O(2^D)$ .

#### RUNNING TIME

The worst case running time of the  $L^3$  algorithm is  $O(m^6 D^3)$ . However in practice, the running time appears to be  $O(mD^3)$ . Also we

might have to make  $O\left(\frac{n}{n-m}\right)^3$  choices of an  $m$ -set of weights before we get a good one. To find the order of the weights we must take  $O(n)$  determinants. Each determinant takes  $O(Y^3)$  multiplications of integers of length  $D$ . So the running time for finding the order is  $O(nY^2D^2)$ . So the total running time on the first part of the algorithm is

$$O\left(m\left(\frac{n}{n-m}\right)^3 D^3 + O(nY^3D^2)\right).$$

The second part of the algorithm is solving for the  $\alpha_i$ 's after a cypher is received. The running time for this part is

$$O(n^2) + O(2^{\epsilon/2}) + O(Y^2).$$

#### TESTS OF THE ALGORITHM

Table 1 summarizes the examples we have run to test the algorithm.

Type - refers to Merkle-Hellman or Graham-Shamir.

N - the number of weights.

R - the number of random bits used in constructing the superincreasing sequence. For MH knapsacks, all random bits are the low order bits. For GS knapsacks, half of the random bits are the low order bits, and the other half are the high order bits.

Y - the number of iterations.

D - the number of bits in the final (or public) weights.

m - the number of weights used in the lattice reduction.

Time - the running time in seconds of the  $L^3$  algorithm on a Cray 1.

Con - Time/(mD<sup>3</sup>)

Table 1.  
Examples

Type	N	Y	R	D	m	Time	Con
GS	40	4	14	82	18	27.6	2.78E-6
MH	40	5	10	93	28	108.2	4.80E-6
MH	40	5	20	113	30	210.6	4.86E-6
MH	40	7	10	105	30	138.1	3.98E-6
MH	40	10	10	123	32	250.1	4.20E-6
MH	50	5	7	97	28	108.7	4.25E-6
GS	50	5	16	96	24	92.3	4.34E-6
MH	50	10	7	127	32	263.2	4.02E-6
GS	64	5	64	158	37	653.7	4.48E-6
GS	64	10	16	140	32	451.8	5.15E-6
GS	100	5	16	151	30	295.5	2.86E-6
GS	100	10	100	270	53	3542.0	3.40E-6
GS	100	20	20	260	52	3355.3	3.67E-6

## CONCLUSION

By extrapolating from the data in Table 1, we project that an iterated knapsack with  $N = 1000$ ,  $Y = 40$ , and  $R = 100$  could be broken in 750 hours on the Cray. Since a knapsack of this size would require a 1.5 Mbit key, it is doubtful that a larger knapsack would ever be seriously considered.

This algorithm will also break Shamir's ultimate knapsack [14]. It appears that the algorithm can be modified to break the knapsack that Brickell presented at Crypto'83 and also the lexicographic knapsack scheme of Petit [12].

## REFERENCES

1. L. Adleman, "On Breaking the Iterated Merkle-Hellman Public Key Cryptosystem," Advances in Cryptology, Proceedings of Crypto 82, Plenum Press 1983, 303-308.
2. E. F. Brickell, "Solving Low-Density Knapsacks," to appear in Advances in Cryptology, Proceedings of Crypto 83, Plenum Press.
3. E. F. Brickell, J. C. Lagarias and A. M. Odlyzko, Evaluation of Adleman's Attack on Multiply Iterated Knapsacks (Abstract), to appear in Advances in Cryptology, Proceedings of Crypto 83, Plenum Press.
4. E. F. Brickell and G. J. Simmons, "A Status Report on Knapsack Based Public Key Cryptosystems," Congressus Numerantium 37 (1983), 3-72.



5. Y. Desmedt, J. Vandewalle, R. Govaerts, "A Critical Analysis of the Security of Knapsack Public Key Algorithms, preprint..
6. J. C. Lagarias, "Simultaneous Diophantine Approximation of Rationals by Rationals, preprint.
7. J. C. Lagarias, "Knapsack Public Key Cryptosystems and Diophantine Approximation," to appear in Advances in Cryptology, Proceedings of Crypto 83, Plenum Press.
8. J. C. Lagarias and A. M. Odlyzko, "Solving Low-Density Subset Sum Problems, Proc. 24th Annual IEEE Symposium on Foundations of Computer Science (1983), 1-10.
9. A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovasz, "Factoring Polynomials with Rational Coefficients," Math. Annalen, 261 (1982), 515-534.
10. R. Merkle and M. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," IEEE Trans. Information Theory IT-24 (1978), 525-530.
11. A. M. Odlyzko, "Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme," preprint.
12. M. Petit, "Etude mathematique de certains systemes de ciphrement: les sacs a dos," doctor's these, Universite de Rennes, France.
13. A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," Proc. 23rd Annual Symposium on Foundations of Computer Science (1982), 145-152.
14. A. Shamir, "The strongest knapsack-based cryptosystem," presented at Crypto 82.

# Dependence of output on input in DES: Small avalanche characteristics

Yvo Desmedt<sup>2</sup>, Jean-Jacques Quisquater<sup>1</sup> and Marc Davio<sup>1,3</sup>

<sup>1</sup> Philips Research Laboratory Brussels,  
Avenue Van Becelaere, 2; Box 8; B-1170 Brussels, Belgium;

<sup>2</sup> Katholieke Universiteit Leuven, Laboratorium ESAT,  
Kardinaal Mercierlaan, 94, B-3030 Heverlee, Belgium;

<sup>3</sup> Université Catholique de Louvain, Batiment Maxwell,  
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium.

**Abstract.** New general properties in the  $S$ -boxes were found. Techniques and theorems are presented which allow to evaluate the non-substitution effect in  $f$  and the key clustering in DES. Examples are given. Its importance related to the security of DES is discussed.

## 1. Introduction

The Data Encryption Standard, in short the DES, is the NBS cryptographic standard for the protection of commercial computer data (FIPS, 1977). Since 1981, it is also an ANSI standard. In the meantime, it is called DEA by ANSI (ANSI, 1980), and it is yet in use in many industrial applications. Recently it has been proposed to become an ISO (International Standard Organisation) standard under the name of DEA1 (ISO, 1983).

There exist several reasons to explore the internal structure and the functional properties in the DES.

1. It can help to understand the DES. Remark that the design criteria of the DES are still classified (Bernhard, 1982).
2. A better understanding of the DES can have two consequences: on the one hand, the detection of weaknesses can speed up a cryptanalysis attack. The detection of inherent strengths will on the other hand simplify the task of defining new standards when they will be needed.
3. The structure can be used in order to simplify or to speed up hardware and software implementations.

To achieve the proposed goals, we first survey (section 2) the technical description of the DES as it appeared in the NBS publication. The reader, who knows the NBS description of the DES, can skip section 2. As the full description of all functions in the DES is very long, we refer to the literature (FIPS, 1977; Konheim, 1981; Meyer & Matyas, 1982; Morris & al., 1977) for these functions.

In section 3 general properties in the  $S$ -boxes and in the key scheduling will be combined.

We analyze several functions in order to combine their properties. As a consequence this can be used to find different cleartexts for which the function  $f$  in the DES gives the same output. These results can also be used to analyze the key clustering in the DES. It means to verify if there exists different keys which gave for most cleartext the same ciphertext.

## 2. NBS description of the DES

The DES algorithm, as described by NBS (FIPS, 1977), consists of three fundamental parts: *enciphering computation*, *calculation of  $f(R, K)$*  and *key scheduling calculation*. They are briefly described below.

First observe that several boxes are used in the DES algorithm. It would be a too long explanation to give the details of all these boxes; it can be found in the NBS description. The kind of boxes (e.g. permutation) will be mentioned. Remark that the input numbering starts from 0 for some boxes and from 1 for the other ones.

In the *enciphering computation*, the input is first permuted by a fixed permutation  $IP$  from 64 bits into 64 bits. The result is split up into the 32 left bits and the 32 right bits, respectively  $L$  and  $R$ . Then a bitwise modulo 2 sum of the left part  $L$  and of  $f(R, K)$  is carried out. After this transformation, the left and right 32 bit blocks are interchanged. Observe that the encryption operation continues iteratively for 16 steps or rounds. In the last round, no interchange of the last obtained left and right parts is performed; the output is obtained by applying the inverse of the initial permutation  $IP$  to the result of the 16<sup>th</sup> round.

In the *calculation of  $f(R, K)$*  the 32 right bits are first expanded to 48 bits in the box  $E$ , by taking some input bits twice, others only once. Then a bitwise modulo 2 sum of the expanded right bits and of 48 key bits is performed. These 48 key bits are obtained in the *key scheduling calculation*, which will be explained later on. The results of the modulo 2 sum go to the eight  $S$ -boxes; each of these boxes has six inputs and four outputs. The  $S$ -boxes are nonlinear functions. The output bits of the  $S$ -boxes are permuted in the box  $P$ .

Let us finally describe the *key scheduling calculation*. The key consists of 64 bits, of which 56 bits only are used. The other 8 bits are not used in the algorithm. The selection of the 56 bits is performed in box  $PC_1$ , together with a permutation. The result is split into two 28 bit words  $C$  and  $D$ . To obtain the 48 key bits for each iteration, the words  $C$  and  $D$  are first left shifted once or twice. A selection and a permutation  $PC_2$  are then applied to the result. The output of  $PC_2$  is the 48 bit key word which is used in  $f(R, K)$ . An additional table tells the user how many shifts must be performed to obtain the next 48 key bits of the key for the following round. The DES can be used in four modes (FIPS, 1980; Konheim, 1981).

### 3. Propagation characteristics

We first analyze the new properties, which we observed in the expansion phase, the  $S$ -boxes and the key scheduling. We combine our results with older ones (Davio, Desmedt & al., 1983) in order to discuss the non-substitution property in  $f$  and the key clustering in the DES. Let us first discuss the importance of the fact that  $f$  is not a substitution and of the key clustering.

#### 3.1. The importance of the propagation characteristics

If  $f$  is not a substitution, for fixed key, the cardinality of the image plays an important role in the evaluation of the security of the DES. Indeed if the image of  $f$  contains only one element, the DES is completely linear. More generally, if the cardinality of the image of  $f$  is small the DES may be insecure.

If there is a key clustering present in the DES, it may be possible that for many cleartexts the effect of modifying the key in a special way does not affect the ciphertext. If this is true for the DES it simplifies enormously an exhaustive attack.

#### 3.2. The expansion phase

The expansion phase plays a very important role in this section.

#### 3.3. The $S$ -boxes

##### 3.3.1. An introduction

We observed several new properties in the  $S$ -boxes. Most of our new properties are valid for all  $S$ -boxes and are consequently called "general properties". In the following sections some of these properties are used in order to analyze in which measure  $f$  is not a substitution and to analyze the key clustering. We did not apply all general properties in the following sections; perhaps in the future one will be able to explain why the  $S$ -boxes have these properties or to use them in some deeper analysis of the DES.

Two kinds of properties are discussed. In the first kind we fix some input bits of the  $S$ -boxes (1, 2, ..., or 5 of the 6 possible bits). We are interested in what changes are propagated at the output and how? E.g. for the output one can wonder if the four output bits are always distinct if we change the non-fixed input bits, or if for some inputs the output is not affected. Secondly we discuss how the output changes if we complement some input bits of the  $S$ -boxes.

We number the inputs of one  $S$ -box by  $abcdef$  as Davies did (Davies, 1981). We number the  $S$ -boxes from 1 to 8 and denote them as  $S_i$ . Remark that representations of the  $S$ -boxes, other than in the NBS norm, may be useful (Davio, Desmedt & al., 1983).

##### 3.3.2. Properties of the $S$ -boxes if some input bits are fixed

The inputs  $a, b, e, f$  of the  $S$ -boxes play a special role in the DES. Indeed one half of the message input bits in each round influences two  $S$ -boxes. These bits will go to the mentioned input bits. These bits will play an important role in the analysis of the non-substitution property of the function  $f$  in the DES. The next properties draw special

attention to the mentioned input bits. The following properties can however easily be generalized. One can easily verify them using a computer program.

We number the properties by a double numbering technique, such that it is easy to refer to them.

1. The observed properties hold for all  $S$ -boxes. We analyze if the output of an  $S$ -box can or cannot change if one modifies the inputs of an  $S$ -box in the following way:

- (a) fix the inputs  $e$  and  $f$ ,
- (b) one is allowed to change  $c$  and  $d$  to an arbitrary value  $c'$  and  $d'$ ,
- (c) one changes the inputs  $a$  and  $b$  as described in the properties,

- 1.1.  $\neg(\forall c, d, c', d', e, f : S_i(0, 0, c, d, e, f) \neq S_i(1, 0, c', d', e, f))$ ,
- 1.2.  $\neg(\forall c, d, c', d', e, f : S_i(0, 1, c, d, e, f) \neq S_i(1, 1, c', d', e, f))$ ,
- 1.3.  $\forall c, d, c', d', e, f : S_i(0, 1, c, d, e, f) \neq S_i(1, 0, c', d', e, f)$ ,
- 1.4.  $\forall c, d, c', d', e, f : S_i(0, 0, c, d, e, f) \neq S_i(1, 1, c', d', e, f)$ .

*Remark:* One can wonder why e.g.  $S_i(0, 0, c, d, e, f)$  was not compared with  $S_i(0, 1, c', d', e, f)$ . This property is already known. Indeed it is known (Konheim, 1981) that each row (see NBS notation) of each  $S$ -box is a permutation. In other words  $S_i(a, b, c, d, e, f) \neq S_i(a, b', c', d', e', f)$  independent of  $b, c, d, e, b', c', d', e'$ . The properties described here are in fact a generalization of it.

2. The observed properties hold for all  $S$ -boxes, except property 2.4. We analyze if the output of an  $S$ -box can or cannot change if one modifies the inputs of an  $S$ -box in the following way:

- (a) fix the inputs  $a$  and  $b$ ,
- (b) one is allowed to change  $c$  and  $d$  to an arbitrary value  $c'$  and  $d'$ ,
- (c) one changes the inputs  $e$  and  $f$  as described in the properties,

- 2.1.  $\neg(\forall a, b, c, d, c', d' : S_i(a, b, c, d, 0, 0) \neq S_i(a, b, c', d', 0, 1))$ ,
- 2.2.  $\neg(\forall a, b, c, d, c', d' : S_i(a, b, c, d, 1, 0) \neq S_i(a, b, c', d', 1, 1))$ ,
- 2.3.  $\neg(\forall a, b, c, d, c', d' : S_i(a, b, c, d, 0, 1) \neq S_i(a, b, c', d', 1, 0))$ ,
- 2.4. If  $i \neq 4$  then:  
 $\neg(\forall a, b, c, d, c', d' : S_i(a, b, c, d, 0, 0) \neq S_i(a, b, c', d', 1, 1))$ .  
 If  $i = 4$  then:  
 $\forall a, b, c, d, c', d' : S_i(a, b, c, d, 0, 0) \neq S_i(a, b, c', d', 1, 1)$ .

*Remark:* The properties 1.3 and 1.4 change if one also allows that the input  $e$  changes to the input  $e'$ . Then it will be possible to find identical outputs for special inputs. A similar remark is true for property 2.4 ( $i = 4$ ) if one allows that the input  $b$  changes.

### 3.3.3. Complementation properties of the $S$ -boxes

A well known (Hellman & al., 1976) property for the  $S$ -boxes is that if one complements one input of an  $S$ -box at least two output bits will change. We analyze the effect of complementing two input bits, while leaving the other ones unchanged. It is evident that one can easily generalize our properties for the case that 3 or more bits are

	<i>ab</i>	<i>ac</i>	<i>ad</i>	<i>ae</i>	<i>af</i>	<i>bf</i>	<i>cf</i>	<i>df</i>	<i>ef</i>
<i>S</i> -box 1	0	6	6	5	0	3	5	2	7
<i>S</i> -box 2	0	4	5	2	0	3	7	1	2
<i>S</i> -box 3	0	3	2	6	5	4	5	3	4
<i>S</i> -box 4	0	8	0	4	0	2	4	2	4
<i>S</i> -box 5	0	1	3	7	0	3	4	6	4
<i>S</i> -box 6	0	3	5	8	1	3	5	5	0
<i>S</i> -box 7	0	7	2	5	2	3	5	3	4
<i>S</i> -box 8	0	5	2	4	0	0	4	2	3

Table 1: shows for how many out of 32 inputs a complementation of two bits of the input of an *S*-box has no effect.

complemented. The first aim was to observe whether it is possible to maintain a constant output if only two bits are complemented. First observe that in order to maintain a fixed output one has to complement bit *a* or *f*, otherwise we conflict with the permutation property of the "rows" in the *S*-boxes. For special *abcdef* inputs the output of an *S*-box remains unchanged if one complements two of the input bits. We give now the results of our research in table 1.

It is remarkable for each *S*-box that if only *ab* is complemented, the output changes. This is however very easy to prove starting from our properties 1.3 and 1.4 of the previous section.

### 3.4. The key scheduling

In our analysis of the key clustering we used in detail the key scheduling in the DES. The ideas of Neutjens about the key scheduling in the DES were very useful in this context (Neutjens, 1983). We now survey them and explain them systematically. We number the 56 key bits from 1 to 64 as in the NBS description (FIPS, 1977).

First of all remark that after  $PC_1$  one can split up the key scheduling in the DES completely in two parts.  $PC_2$  does not affect this decomposition (Davio, Desmedt & al., 1983). As a consequence of this decomposition, one can separate for one round in the DES the selection of the key bits which will influence the first four *S*-boxes and the last four *S*-boxes. Let us now construct the equivalent scheme. All used notations, e.g. the registers *C* and *D*, originate from the NBS representation of the DES.

We represent the register content of *C* by  $(c_1, c_2, \dots, c_{28})$  and that of *D* by  $(d_1, d_2, \dots, d_{28})$ . Mostly in the key scheduling the registers *C* and *D* are shifted *twice* to obtain the  $K_i$  of the  $i^{\text{th}}$  round, e.g.  $(c_1, c_2, c_3, \dots, c_{28})$  is transformed into  $(c_3, c_4, c_5, \dots, c_2)$ . This can now be reformulated for the *C* register as *one* shift on the following *two* registers  $(c_1, c_3, c_5, \dots, c_{27})$  and  $(c_2, c_4, c_6, \dots, c_{28})$ . We call them respectively the odd and the even registers. One can then realize the key scheduling with 4 registers instead of two, which shift only once when in the NBS representation the registers shift twice. This reorganization affects the  $PC_2$ .

One has now still to discuss what happens if only one shift is performed on *C* and *D* as in the iterations 1, 2, 9 and 16 using our equivalent representation. The first shift in the first iteration can be realized together with  $PC_1$ . In the other situations we interchange the content of the odd and the even registers, by performing first a shift on the old content

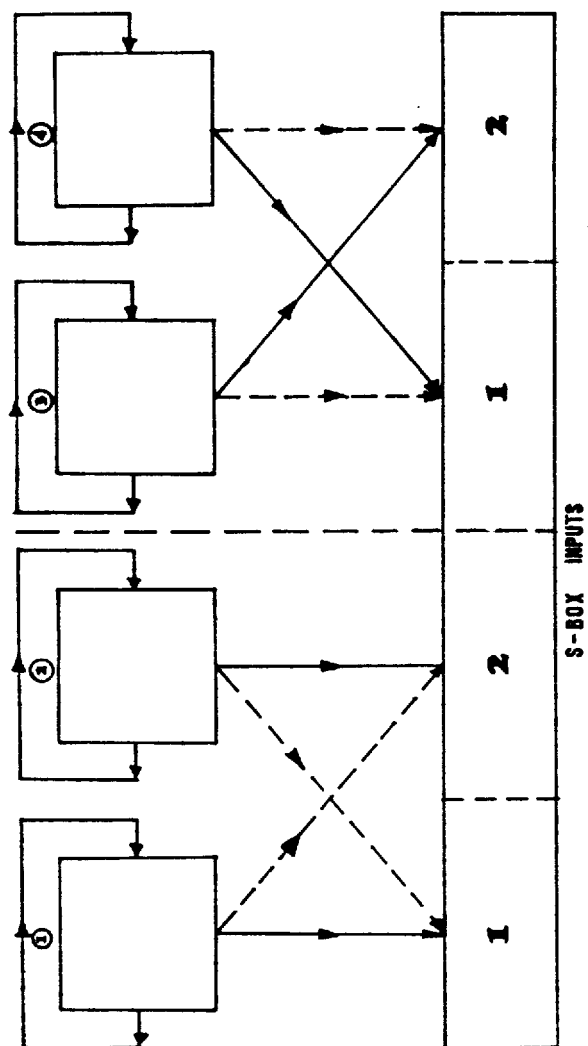


Figure 1: An equivalent key scheduling.

of the odd register and no shift on that of the even register. We then change also the name of each register: odd becomes even, even becomes odd. Indeed  $(c_1, c_3, c_5, \dots, c_{27})$ ,  $(c_2, c_4, c_6, \dots, c_{28})$  is then changed into  $(c_2, c_4, c_6, \dots, c_{28})$ ,  $(c_3, c_5, c_7, \dots, c_1)$ . One can verify that previous operations are identical to one shift in the NBS notation.

The register  $D$  can be treated in a similar way. Remark that it is more difficult to perform one shift in the NBS representation. However we are able to see better which bits of the key affect a particular  $S$ -box. We now represent this result in tables 2-5 and fig. 1, where  $X$  means that this key bit is not selected by  $PC_2$ .

Let us now apply all the described properties.

### 3.5. The function $f$ is not one-to-one for fixed $K$

Let us remember here that the function  $f$  consists of the expansion box  $E$ , of the EXOR-ing with the key bits, of the  $S$ -boxes and of the permutation  $P$ . It has sometimes been wondered whether the  $f$  function is by itself a substitution. The answer to that question is negative (Davio, Desmedt & al., 1983; Konheim, 1981). A more systematic discussion is given in this section.

We will now use the properties described in section 3.3.2. to demonstrate how they can be used in the analysis of the non-substitution of the function  $f$ . Evidently we assume that the key  $K$  is fixed. We analyze which bits of the message part  $R$  (see NBS notation) one must change in order to maintain the same output of the function  $f$ . We will progressively increase the number of changed bits. First we only change the inputs (or message part of the input) of one, two and then three  $S$ -boxes and generalize afterwards. We will mostly use the new as well as the well known (Hellman & al., 1976; Konheim, 1981) general properties of the  $S$ -boxes, together with the structure of  $E$  (Davio, Desmedt & al., 1983).

**Theorem 1:** If for fixed key, one only changes the input of one  $S$ -box the output of the function  $f$  will change.

**Proof:** In order not to affect the inputs of the other  $S$ -boxes one can only change the inputs  $c$  and  $d$ . However if the inputs  $a$  and  $b$  are not changed an  $S$ -box forms a substitution. ■

**Theorem 2:** If for fixed key, one changes only the input of two neighbourhood  $S$ -boxes the output of the function  $f$  will change.

**Proof:** Let us call the two affected  $S$ -boxes,  $S_i$  and  $S_{i+1}$  and let us define  $S_0$  as being  $S_1$  (this again shows that it can be more interesting to start the numbering from 0, see (Davio, Desmedt & al., 1983)). In order not to affect the input of  $S_{i-1}$  the inputs  $a$  and  $b$  of  $S_i$  may not change and similarly for the inputs  $e$  and  $f$  of  $S_{i+1}$  in order not to affect the inputs of  $S_{i+2}$ . In order not to conflict with the permutation properties of the "rows" of the  $S$ -boxes and using the previous remark, at least the input  $f$  in  $S_i$  must be complemented in order to maintain a fixed output. A similar remark is true for the input  $a$  of  $S_{i+1}$ . As consequence of the expansion box  $E$  a complementation of the input  $e$  (respectively  $f$ ) of  $S_i$  is equal to a complementation of the input of  $a$  (respectively  $b$ ) of  $S_{i+1}$ . So in order to produce a same output we have at least to complement  $a$  and  $b$  in  $S_{i+1}$ . Remark that the inputs  $c$  and  $d$  in  $S_{i+1}$  do not influence the proof. In other words



	3	23	9	2	14	11	13	X2	21	5	7	6	20	X3
1	34	18	2	51	35	19	3	52	36	49	33	17	1	50
2	26	10	59	43	27	11	60	44	57	41	25	9	58	42
3	10	59	43	27	11	60	44	57	41	25	9	58	42	26
4	59	43	27	11	60	44	57	41	25	9	58	42	26	10
5	43	27	11	60	44	57	41	25	9	58	42	26	10	59
6	27	11	60	44	57	41	25	9	58	42	26	10	59	43
7	11	60	44	57	41	25	9	58	42	26	10	59	43	27
8	60	44	57	41	25	9	58	42	26	10	59	43	27	11
9	52	36	49	33	17	1	50	34	18	2	51	35	19	3
10	36	49	33	17	1	50	34	18	2	51	35	19	3	52
11	49	33	17	1	50	34	18	2	51	35	19	3	52	36
12	33	17	1	50	34	18	2	51	35	19	3	52	36	49
13	17	1	50	34	18	2	51	35	19	3	52	36	49	33
14	1	50	34	18	2	51	35	19	3	52	36	49	33	17
15	50	34	18	2	51	35	19	3	52	36	49	33	17	1
16	42	26	10	59	43	27	11	60	44	57	41	25	9	58

Table 2: The effect of the selection of the key bits (1-64) by  $PC_1$  and  $PC_2$ . The first row of the table indicates to which input of the  $S$  boxes the key bits go. (Neutjens, 1983)

	4	17	8	24	16	10	18	12	15	1	19	X1	22	X4
1	60	44	57	41	25	9	58	42	26	10	59	43	27	11
2	52	36	49	33	17	1	50	34	18	2	51	35	19	3
3	36	49	33	17	1	50	34	18	2	51	35	19	3	52
4	49	33	17	1	50	34	18	2	51	35	19	3	52	36
5	33	17	1	50	34	18	2	51	35	19	3	52	36	49
6	17	1	50	34	18	2	51	35	19	3	52	36	49	33
7	1	50	34	18	2	51	35	19	3	52	36	49	33	17
8	50	34	18	2	51	35	19	3	52	36	49	33	17	1
9	42	26	10	59	43	27	11	60	44	57	41	25	9	58
10	26	10	59	43	27	11	60	44	57	41	25	9	58	42
11	10	59	43	27	11	60	44	57	41	25	9	58	42	26
12	59	43	27	11	60	44	57	41	25	9	58	42	26	10
13	43	27	11	60	44	57	41	25	9	58	42	26	10	59
14	27	11	60	44	57	41	25	9	58	42	26	10	59	43
15	11	60	44	57	41	25	9	58	42	26	10	59	43	27
16	3	52	36	49	33	17	1	50	34	18	2	51	35	19

Table 3: Similar as table 2. (Neutjens, 1983)

	34	29	38	33	42	30	47	27	35	X5	28	39	25	X8
1	53	37	21	5	20	4	55	39	23	7	54	38	22	6
2	45	29	13	28	12	63	47	31	15	62	46	30	14	61
3	29	13	28	12	63	47	31	15	62	46	30	14	61	45
4	13	28	12	63	47	31	15	62	46	30	14	61	45	29
5	28	12	63	47	31	15	62	46	30	14	61	45	29	13
6	12	63	47	31	15	62	46	30	14	61	45	29	13	28
7	63	47	31	15	62	46	30	14	61	45	29	13	28	12
8	47	31	15	62	46	30	14	61	45	29	13	28	12	63
9	39	23	7	54	38	22	6	53	37	21	5	20	4	55
10	23	7	54	38	22	6	53	37	21	5	20	4	55	39
11	7	54	38	22	6	53	37	21	5	20	4	55	39	23
12	54	38	22	6	53	37	21	5	20	4	55	39	23	7
13	38	22	6	53	37	21	5	20	4	55	39	23	7	54
14	22	6	53	37	21	5	20	4	55	39	23	7	54	38
15	6	53	37	21	5	20	4	55	39	23	7	54	38	22
16	61	45	29	13	28	12	63	47	31	15	62	46	30	14

Table 4: Similar as table 2. (Neutjens, 1983)

	32	44	37	43	36	45	26	X6	40	31	48	41	46	X7
1	30	14	61	45	29	13	28	12	63	47	31	15	62	46
2	22	6	53	37	21	5	20	4	55	39	23	7	54	38
3	6	53	37	21	5	20	4	55	39	23	7	54	38	22
4	53	37	21	5	20	4	55	39	23	7	54	38	22	6
5	37	21	5	20	4	55	39	23	7	54	38	22	6	53
6	21	5	20	4	55	39	23	7	54	38	22	6	53	37
7	5	20	4	55	39	23	7	54	38	22	6	53	37	21
8	20	4	55	39	23	7	54	38	22	6	53	37	21	5
9	12	63	47	31	15	62	46	30	14	61	45	29	13	28
10	63	47	31	15	62	46	30	14	61	45	29	13	28	12
11	47	31	15	62	46	30	14	61	45	29	13	28	12	63
12	31	15	62	46	30	14	61	45	29	13	28	12	63	47
13	15	62	46	30	14	61	45	29	13	28	12	63	47	31
14	62	46	30	14	61	45	29	13	28	12	63	47	31	15
15	46	30	14	61	45	29	13	28	12	63	47	31	15	62
16	38	22	6	53	37	21	5	20	4	55	39	23	7	54

Table 5: Similar as table 2. (Neutjens, 1983)

even if one additionally changes the inputs  $c$  and  $d$  in  $S_{i+1}$  or does not, the output of  $S_{i+1}$  will change, by virtue of property 1.3 and 1.4 of the  $S$ -boxes. ■

**Theorem 3:** Assume that for fixed key one changes only the input of three neighbouring  $S$ -boxes, the output of the function  $f$  will for some inputs remain identical only if at least all of the following conditions are satisfied together:

1. one complements the inputs  $a, b$  and  $e$  of the middle of the three  $S$ -boxes,
2. one complements the input  $c$  or  $d$  of the last  $S$ -box,
3. one does not complement the input  $f$  of the middle of the three  $S$ -boxes.

*Proof:* We call the three  $S$ -boxes  $S_{i-1}$ ,  $S_i$  and  $S_{i+1}$  where  $S_0$  is equal to  $S_8$  and  $S_9$  equals  $S_1$ . The proof is for a large part similar to that of theorem 2. Let us first give the similar part of the proof.

We must fix the inputs  $a$  and  $b$  of  $S_{i-1}$ , and  $e$  and  $f$  of  $S_{i+1}$ . The input  $f$  of  $S_{i-1}$  must be complemented and similarly for the input  $a$  of  $S_{i+1}$ . This last condition is equivalent to say that the inputs  $b$  and  $e$  of  $S_i$  must be complemented. Now we apply the consequences of theorem 2 to continue our proof.

If  $a$  and  $b$  are both complemented in  $S_{i+1}$ , the output will change (see proof of theorem 2 or properties 1.3 and 1.4 of the  $S$ -boxes). Using previous observations the input  $b$  in  $S_{i+1}$  may not be complemented, or equivalently the input  $f$  in  $S_i$ . At this moment we already know that for  $S_i$  the inputs  $b$  and  $e$  must be complemented and  $f$  may not. Because each row in the  $S$ -boxes is a permutation and because the input  $f$  may not be complemented in  $S_i$ , the input  $a$  must be complemented in  $S_i$ . Remark that in fact one must still complement input  $c$  or  $d$  in  $S_{i+1}$ . Indeed if only one input bit in an  $S$ -box is complemented, the output changes. ■

We have now proven the theorem. It is now very easy to generate in a systematic way several examples for which the function  $f$  remains constant even if some bits are complemented.

### 3.6. The key clustering

We analyze the clustering from the point of view that the DES contains  $j$  rounds, where  $j$  is between 1 and 16. The input for these  $j$  rounds is fixed, while we complement or change some bits of the key. So if we speak now about an input of an  $S$ -box, this input is related to a modification of the key.

We first prove some general theorems for the key clustering, and afterwards we give some examples.

#### 3.6.1. A general approach

First of all for a fixed input the permutation  $IP$  has no influence on the key clustering. We can start the analysis from  $L_s$  and  $R_s$ . This means that if we are interested in a complete DES analysis  $s = 0$  and  $j = 16$ . Let us now apply the DES with the key  $K$  and

$K'$  and call the subkeys  $K_1$  till  $K_{16}$  and  $K'_1$  till  $K'_{16}$ . The key  $K$  will produce some  $L$  and  $R$  register content, while  $K'$  produces  $L'$  and  $R'$ . The effect of the first of the  $j$  rounds is that in the case we use the key  $K$  we have  $L_{s+1} = R_s$  and  $R_{s+1} = L_s \oplus f(R_s, K_{s+1})$ . Applying the key  $K'$  we obtain  $L'_{s+1} = R_s$  and  $R'_{s+1} = L_s \oplus f(R_s, K'_{s+1})$ . After  $t$  rounds we obtain using key  $K$  the register content  $L_{s+t} = R_{s+t-1}$  and  $R_{s+t} = L_{s+t-1} \oplus f(R_{s+t-1}, K_{s+t})$ . Using the key  $K'$  we have:  $L'_{s+t} = R'_{s+t-1}$  and  $R'_{s+t} = L'_{s+t-1} \oplus f(R'_{s+t-1}, K'_{s+t})$ . Remark that in general by changing the key the content of the registers  $L$  and  $R$  change too. Let us now call  $H_{s+t} = f(R_{s+t-1}, K_{s+t}) \oplus f(R'_{s+t-1}, K'_{s+t})$ . It is now easy to see using (Davio, Desmedt & al., 1983) that the global effect of a change in the key has no final effect on the ciphertext if the two following conditions are satisfied together.

1.  $H_{s+1} \oplus H_{s+3} \oplus H_{s+5} \oplus \dots \oplus H_t = 0$ , where  $t = s + j$  if  $j$  is odd, else  $t = s + j - 1$ .
2.  $H_{s+2} \oplus H_{s+4} \oplus H_{s+6} \oplus \dots \oplus H_u = 0$ , where  $u = s + j$  if  $j$  is even, else  $u = s + j - 1$ .

Using previous conditions it is now easy to analyze the conditions necessary for key clustering if one analyzes only 1, 2, 3 or 4 rounds. The analyze of more rounds seems to be more difficult.

### 3.6.2. An analysis of the key clustering in a DES with 1, 2, 3 or 4 rounds

In the case one round is considered we must have  $H_{s+1} = 0$ . This means  $f(R_s, K_{s+1}) = f(R_s, K'_{s+1})$ . Using previous knowledge on the  $S$ -boxes this means that the input of an  $S$ -box is not changed or that at least two bits change. It is very easy to generate several examples for this case. Using the fact that  $E$  is an expansion of 32 bits to 48 bits and its structure (Davio, Desmedt & al., 1983) and because  $PC_2$  selects only 48 bits out of the 56 bits of the key we have the following result. *For each (cleartext, ciphertext) pair in a one round DES there exist exactly  $2^{24}$  keys which generate the same (cleartext, ciphertext) pair starting from a fixed cleartext.* If a similar remark remains true for the complete DES algorithm (16 rounds), the DES is very easy to break using a simplified exhaustive attack. Let us therefore start to analyze more rounds.

In the case two rounds are considered we must have  $H_{s+1} = 0$  and  $H_{s+2} = 0$ . This means  $f(R_s, K_{s+1}) = f(R_s, K'_{s+1})$ , as in previous case, and additionally  $f(R_{s+1}, K_{s+2}) = f(R_{s+1}, K'_{s+2})$ , because from the first equality we have  $R'_{s+1} = R_{s+1}$ . Remark that the  $S$ -boxes must satisfy similar conditions as in the case only one round was considered. However to satisfy it for the two rounds together we must take the key scheduling in the DES into consideration. This is now easy to do if one uses the tables explained earlier. We now give a simple example of it.

**Example 1.** If one complements the bits 3 and 44 (in the NBS notation) of any 64 bit key, then there exists  $6 \cdot 2^{59}$  pairs of (cleartext, ciphertext) which remain identical during round 1 and 2 in the DES. In other words, about 1/5 of all pairs (cleartext, ciphertext) are not affected by the complementation of 2 bits of the key, during round 1 and 2.

Let us now explain using fig. 2 what happens and how one can calculate the (cleartext, ciphertext) pairs. The bits 3 and 44 go both after the key scheduling in the first round to  $S_3$  and become there the inputs  $a$  and  $e$ . Using table 1 we know that for 6 out of 32 (or 12 out of 64) possible inputs a complementation of  $a$  and  $e$  in  $S_3$  does not change the output. This means that the possible inputs for which the above property is true are restricted from  $2^{64}$  to  $6 \cdot 2^{59}$ . The cardinality of the set of cleartexts for which the explained clustering is satisfied is independent of the used key. However the set of cleartexts for which the above clustering is satisfied, changes if other keys are considered. This is a consequence of the

**Necessary :**

$S_3$  not influenced by complementing 3 and 44

$\Rightarrow$  table

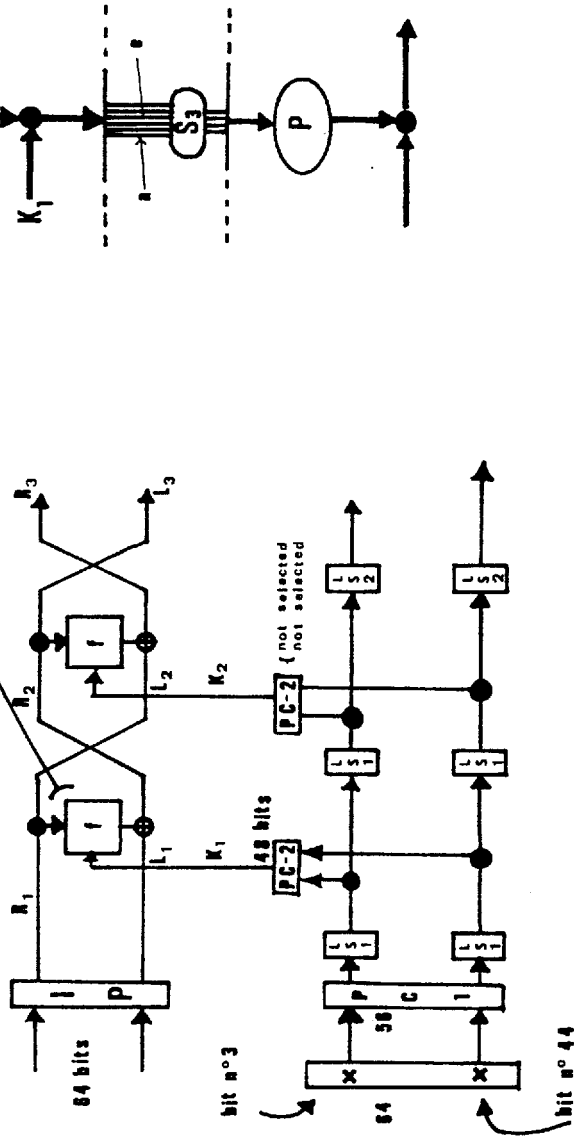


Figure 2: Example 1 on the key clustering in a two round DES.

input $S_3$ <i>abcdef</i>	output $S_3$
000100	1001
100110	idem
000101	0000
100111	idem
010000	0001
110010	idem
010100	1100
110110	idem
010111	1110
110101	idem
011011	1011
111001	idem

Table 6: Inputs (in binary form) for  $S_3$  which generate the same output if the bits  $a$  and  $e$  are complemented.

exor of the subkey with the expanded  $R$  register in the function  $f$ . Let us now analyze which input for the  $S$ -boxes we must force in order to satisfy the key clustering. The input for  $S_3$ , in the first round, must be one of those collected in table 6, in order to satisfy the key clustering. Now we must still analyze which restrictions the second round imposes on the possible cleartext. The analysis in this example is straightforward because the key bits 3 and 44 are not selected in the second round, so no extra condition is necessary.

One may observe that we were lucky in the construction of the previous example. First the non-selection of the key bits in the second iteration seems to be lucky. Secondly example 1 is only valid for rounds 1 and 2 in the DES. In the following example the reader can observe that similar examples can be given for all rounds and that it is not necessary that some key bits are not selected in the second or first round.

*Example 2.* This example is true for most consecutive rounds. As a consequence of the ideas of Neutjens on the key scheduling (see section 3.4), two consecutive rounds can mostly be analyzed systematically. (Neutjens, 1983). This is true if one uses two shifts in the key scheduling, as represented by the NBS, to move to the next round. This means the rounds 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 9-10, 10-11, 11-12, 12-13, 13-14 and 14-15. In order not to affect the generality we will use a more general descriptions of the property. If one complements the two bits of the key which will "arrive" in  $S$ -box 4 at locations  $a$  and  $e$  during the first of the two above rounds, then for every key there exists  $24 \cdot 2^{54}$  (or about  $1/43$  of all possible) pairs (cleartext, ciphertext) which remain identical during two consecutive rounds mentioned earlier. This can be easily analyzed (similar as in example 1) using tables 2-5, and using our properties of the  $S$ -boxes (table 1).

Let us now consider three consecutive rounds. First more restrictions on the cleartext are then imposed in order not to affect the ciphertext if one modifies the key. This is a consequence of the key scheduling. However the output of the function  $f$  in the first and last (of the three) rounds must no longer be constant (see section 3.6.1). This relaxes the imposed restrictions. Let us give a short example to illustrate it.

*Example 3.* The three consecutive rounds may be 2-3-4, 3-4-5, 4-5-6, 5-6-7, 6-7-8,

9-10-11, 10-11-12, 11-12-13, 12-13-14 and 13-14-15. Hereto one complements (e.g.) three bits of the key (fig. 3). In our example the three key bits must "arrive" at location  $a$  and  $d$  in  $S$ -box 8 in the first round (of the three consecutive) and at location  $d$  in  $S$ -box 4 in the second round (of the three consecutive). We call these three key bits respectively  $k_1$ ,  $k_2$  and  $k_3$ . By analyzing the box  $P$  (see (Davies, 1981)) and using section 3.6.1 two cases can be distinguished.

1. The third output bit of  $S_8$  is complemented in the first and third iteration (of the three consecutive) as a consequence of the previous modification of the key. In other words bit 15 of the output of  $f$  (after the box  $P$ ) must be complemented in the first and last round. The modification of the previous bit will have no influence at all in the second round of the three. Indeed after the expansion phase it is exored with key bit  $k_3$  which we complemented too. Remark first that *the set of cleartexts for which the above clustering is satisfied changes if other keys are considered*. This is a consequence of the xor of the subkey with the expand  $R$  register in the function  $f$ . Let us now analyze which input we must force at *the input of the  $S$ -boxes*, in the three rounds, in order to satisfy the above conditions. Remember from Fig. 2 that the input of the  $S$ -boxes is equal to the subkey xor the expanded  $R$  register. In the first round key bits  $k_1$  and  $k_2$  influence respectively the input  $a$  and  $d$  in  $S_8$ , as a consequence of our choice.  $k_3$  is not selected. The input of  $S_8$  must be chosen from table 7. In the second round (of the three consecutive) we yet discussed the influence of key bit  $k_3$ . Using table 2-5 we find that  $k_1$  and  $k_2$  become now the input  $a$  and  $e$  respectively in  $S_7$ . The input of  $S_7$  must be chosen from table 8. In the third round  $k_1$  and  $k_2$  influence respectively the inputs  $b$  and  $f$  from  $S_8$ . The input of  $S_8$  must be chosen from table 9.
2. The second and third output bits of  $S_8$  are complemented in the first and third round as a consequence of the previous modification of the key. We must then choose the inputs of  $S_8$  in the first round out of table 10, the inputs of  $S_7$  in the second round out of table 11 and the input of  $S_8$  in the third round out of table 12. This can be analyzed in a similar way as for the first case.

We can then analyze that for 50% of the keys: For 21 on 16384 (about 1/780) cleartexts, the ciphertext is not modified. For the other 50% of the keys this happens for 1 on 2048 cleartexts. This analysis is involved. The reader can check it using tables 7-12. He must then take into consideration that the tables impose conditions on the cleartext input of the three rounds. Using fig. 3 he can then easily prove that the first round imposes some conditions on the right input of the cleartext. Similarly the second round imposes some conditions on the cleartext at the left side input of the three rounds. To analyze the restrictions on the input as a consequence of the third round the reader must use the property that each round is a substitution from  $2^{64}$  to  $2^{64}$  elements (Davio, Desmedt & al., 1983) for fixed key. Care should be taken in performing this last step. It is possible that previously imposed conditions influence the new one. Indeed by imposing special conditions on the cleartext, some restrictions can exist on the output of  $f$  in previous rounds.

Other examples can easily be generated. It would be interesting to generalize the previous examples to the complete DES with 16 rounds.

we complement only 3 bits  
of the key

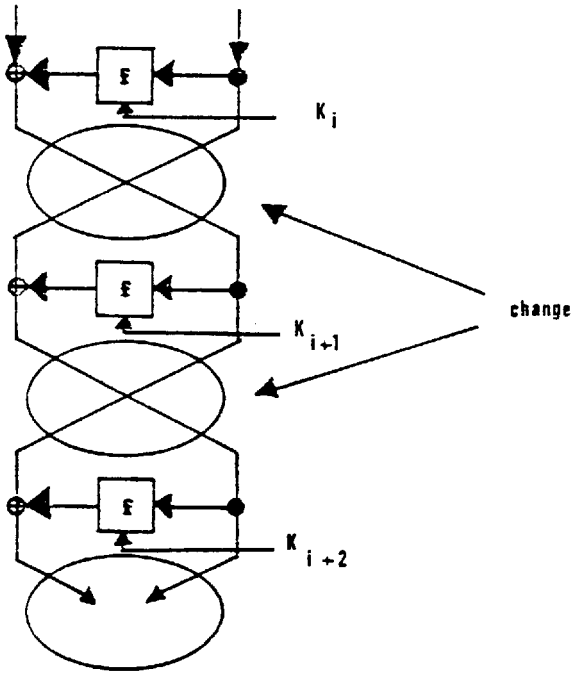


Figure 3: The key clustering in a three round DES.



input $S_8$ <i>abcdef</i>	output $S_8$
001001	1010
101101	1000
001100	1011
101000	1001

Table 7: Inputs (in binary form) for  $S_8$  which generate outputs in which the third output bit is complemented if the bits  $a$  and  $d$  of the input are complemented.

input $S_7$ <i>abcdef</i>	output $S_7$
000000	0100
100010	idem
001001	0100
101011	idem
001111	1010
101101	idem
011000	0101
111010	idem
011001	0010
111011	idem

Table 8: Inputs (in binary form) for  $S_7$  which generate the same output if the bits  $a$  and  $e$  are complemented.

input $S_8$ <i>abcdef</i>	output $S_8$
001100	1011
011101	1001
110000	0000
100001	0010

Table 9: Inputs (in binary form) for  $S_8$  which generate outputs in which the third output bit is complemented if the bits  $b$  and  $f$  of the input are complemented.

input $S_8$ <i>abcdef</i>	output $S_8$
011001	0000
111101	0110

Table 10: Inputs (in binary form) for  $S_8$  which generate outputs in which the second and third output bit is complemented if the bits  $a$  and  $d$  of the input are complemented.

input $S_7$ <i>abcdef</i>	output $S_7$
000010	1011
100100	idem
000101	1011
100011	idem
010101	0101
110011	idem

Table 11: Inputs (in binary form) for  $S_7$  which generate the same output if the bits  $a$ ,  $d$  and  $e$  are complemented.

input $S_8$ <i>abcdef</i>	output $S_8$
001000	0110
011001	0000
000011	1111
010010	1001
000111	1000
010110	1110

Table 12: Inputs (in binary form) for  $S_8$  which generate outputs in which the second and third output bits are complemented if the bits  $b$  and  $f$  of the input are complemented.

## 4. Conclusions and perspectives

A cryptographic system can only be considered secure if a small modification in the cleartext and/or in the key strongly affect on a non-linear way the ciphertext. We described techniques for analyzing this constraint for the DES. We found that if the DES had only a few rounds it would be a weak system. Our analysis demonstrated at the same time that the known probabilistic test done on the DES are insufficient to conclude that the scheme is secure. Were it possible to work out on a 16-round DES the techniques presented here one could possibly prove the so often alleged existence of a key clustering in the DES.

### References

- ANSI X3.92-1981, "Data Encryption Algorithm," American National Standards Institute, New York (December 31, 1980).
- Ayoub, F., "On the design of SP-networks," presented at Eurocrypt '83, 21-23 March 83, Udine, Italy.
- Bernhard, R., "Breaching system security," *Spectrum*, vol. 19, pp. 24-31 (1982).
- Davies, D. W., "Some regular properties of the Data Encryption Standard algorithm," NPL note, presented at Crypto '81 (1981).
- Davio, M., Desmedt, Y., Fosséprez, M., Govaerts, R., Hulsbosch, J., Neutjens, P., Piret, P., Quisquater, J. J., Vandewalle, J. & Wouters, P., "Analytical characteristics of the DES," pp. 171-202, in *Advances in cryptology: Proc. of CRYPTO '83*, Santa Barbara, ed. D. Chaum, Plenum Publishing Corp., New York (1984).
- Denning, D. E., *Cryptography and data security*, Addison Wesley, Reading (Mass.) (1982).
- Diffie, W. & Hellman, M. E., "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, n° 6, pp. 74-84 (1977).
- FIPS publication 46, "Data Encryption Standard," Federal Information Processing Standard, National Bureau of Standards, U.S. Department of Commerce, Washington, D.C. (January 1977).
- Fosséprez, M. & Wouters, P., "Cryptanalyse et matérialisation des réseaux de chiffrement," Final work, Université Catholique de Louvain, Belgium (1983).
- Hellman, M. E., Merkle, R., Schroepel, R., Washington, L., Diffie, W., Pohlig, S. & Schweitzer, P., "Results of an initial attempt to cryptanalyze the NBS data encryption standard," SEL 76-042, Stanford University (1976).
- Hulsbosch, J., "Analyse van de zwakheden van het DES-algoritme door middel van formele codering," Final work, Katholieke Universiteit Leuven, Belgium (1982).
- ISO/DP 8227 (Draft proposal), "Data encipherment, specification of algorithm DEA1," (1983).
- Konheim, A. G., *Cryptography: A primer*, J. Wiley, New York (1981).
- Morris, R., Sloane, N. J. A. & Wyner, A. D., "Assessment of the NBS proposed Data Encryption Standard," *Cryptologia*, vol. 1, pp. 301-306 (1977).
- Neutjens, P., "Diepere inzichten en eenvoudige hardware voor DES cryptografisch algoritme aan de hand van equivalente structuren," Final work, Katholieke Universiteit Leuven, Belgium (1983).

# DES HAS NO PER ROUND LINEAR FACTORS

J. A. Reeds and J. L. Manferdelli

AT&T Bell Laboratories  
Murray Hill, New Jersey 07974

## ABSTRACT

Interest in the cryptanalysis of the National Bureau of Standards' Data Encryption Standard (DES) has been strong since its announcement. Here we describe an attack on a class of ciphers like DES based on linear factors.

If DES had any non trivial factors, these factors would provide an easier attack than one based on complete enumeration. Basically, a factor of order  $n$  reduces the cost of a solution from  $2^{56}$  to  $2^n + 2^{56-n}$ . At worst ( $n=1$  or  $55$ ), this reduces the cost of a Diffie-Hellman search machine from 20 million dollars to 10 million dollars: a 10 million dollar savings. At best ( $n=28$ ), even without iteration, the method could reduce the cost from  $2^{56}$  to  $2^{28} + 2^{28}$ : a computation well within the reach of a personal computer.

Alas, DES has no such linear factors.

## INTRODUCTION

The basic idea here is an elaboration of a trivial idea, too good to be true. If, for each distinct value of the key, DES mapped the plaintext blocks into the ciphertext blocks *linearly*, one could deduce the matrix of that linear transformation from a small number of corresponding plaintext/ciphertext blocks. Similarly, if the dependence of the ciphertext on the key was linear, one could solve for the key. Unfortunately, the S boxes introduce strong nonlinearities: each bit output from each S box can only be represented by polynomials (in 6 variables) over GF(2) with many terms (for a discussion of these representations and their connection to coding theory see [2], chapters 2,13,14).

The current elaboration is that there might be three special linear functions of the plaintext, ciphertext and the key respectively such that the mapped ciphertext depends only on the mapped plaintext and mapped key. If the mapped key has lower dimensionality than the unmapped key, one can attempt to solve the mapped cryptosystem (possibly by brute force search).

This mapping behavior is called *cryptosystem factorization*. In general, a cryptosystem consists of a plaintext space, a key space, a ciphertext space, and a family of invertible maps indexed by the key space. We say that cryptosystem A is a *factor* of cryptosystem B if there are maps (called *factor maps*) between the plaintext, key, and ciphertext spaces such that the enciphering and deciphering actions of cryptosystem A can be recovered from those of cryptosystem B using the factor maps. If the factor mappings are linear functions we say A is a *linear factor* of B. If the key space of A is smaller than that of B one can profitably break B by first breaking A.

There is no special reason to suppose that the DES has any factors, linear or not. But if it had they probably would have the same general round-by-round flavor that DES itself has. This paper shows that the individual round of DES has no linear factors.

## DES NOTATION

DES is a product cipher. The key dependent transformation that DES induces on the plaintext is a product of a family of (involutory) transformations  $\mu$  and  $\lambda_i$ . If  $L$  and  $R$  are the two 32 bit subwords of a 64 bit input, we have\*

$$\mu : LR \mapsto RL$$

and

$$\lambda_i : LR \mapsto L(R + f(E(L) + k_i))$$

Using these conventions,

$$DES(K, P) = IP^{-1} \lambda_{16} \mu \lambda_{15} \mu \cdots \lambda_2 \mu \lambda_1 IP(LR)$$

and,

$$DES^{-1}(K, P) = IP^{-1} \lambda_1 \mu \lambda_2 \cdots \lambda_{15} \mu \lambda_{16} IP(LR).$$

\* The sign "+" in this paper denotes addition. Here, we do addition in at least three different rings: the ordinary integers  $[1+1=2]$ ,  $GF(2)$   $[1+1=0]$ , and vector spaces over  $GF(2)$   $[(1,0,1,1)+(1,1,0,1)=(0,1,1,0)]$ . To emphasize that we are interested in the arithmetical properties of the "+" operator, we use + in all three cases. We rely on the reader to distinguish which ring (and hence which operator) is being used in any given equation. In the second displayed equation, for example, the first plus denotes addition done in the vector space of dimension 32 over  $GF(2)$ ; the second plus refers to arithmetic done in the vector space of dimension 48 over  $GF(2)$ . Readers who are not familiar with DES will see in a few paragraphs why the rings in the second equation are what we say they are.

The transformation  $IP$  consists of a permutation of the input bits; it has no cryptographic significance and need not be mentioned any further.  $E$ ,  $P$  and the  $S$  boxes  $S_1, \dots, S_8$  are defined in [1] and will be discussed in more detail below.  $k_i$  ( $i=1,2,3,\dots,16$ ) is a 48 bit subkey for round  $i$  derived from a 56 bit key  $k$  according to a key schedule described in [1]. We refer to the composed map  $\sigma_i = \mu\lambda_i$  as a "round" of DES. Note that DES is composed of 16 encrypting rounds with the switch of the 32 bit subwords suppressed in the final round.

Denoting the vector space of dimension  $n$  over  $GF(2)$  by  $V_n$ , we have:

$$E: V_{32} \longrightarrow V_{48}$$

$$P: V_{32} \longrightarrow V_{32}$$

$$f: V_{48} \longrightarrow V_{32}$$

$$S_i: V_6 \longrightarrow V_4$$

$E$  is the expansion matrix which takes  $x = (x_1, \dots, x_{32})$  to  $(x_{E(1)}, \dots, x_{E(48)})$ . The function  $f$  is obtained by applying successive  $S$  boxes to the successive six bits of the argument and then applying the permutation matrix  $P$  to the resultant vector, i.e.:

$$y = (S_1(x_1, \dots, x_6), \dots, S_8(x_{43}, \dots, x_{48}))$$

$$f(x) = (y_{P(1)}, \dots, y_{P(32)})$$

$E$  and  $P$  are linear functions,  $f$  is not. Writing this in tabular form, we get

Two rounds of DES		
round	left 32 bits	right 32 bits
0	$L$	$R$
1	$R$	$L + f(E(R) + k_1)$
2	$L + f(E(R) + k_1)$	$R + f(E[L + f(E(R) + k_1)] + k_2)$

It is convenient to employ another set of equations to describe DES. Setting  $x_0 = E(L)$ ,  $x_1 = E(R)$  and  $x_2 = E(L + f(E(R) + k_1))$ , we can write a recurrence based on the second column of the table above.

$$x_0 + x_2 = \phi(x_1 + k_1)$$

where

$$\phi(x) = Ef(x) \quad (1)$$

In fact, if we write down all 16 rounds (and perform an extra switch of the two 32 bit subwords at the end), we see that

$$x_{i+1} + x_{i-1} = \phi(x_i + k_i) \quad (2)$$

given the obvious definition for  $x_i$  for  $i = 1, 2, \dots, 16$ . With this notation, the output of the DES algorithm consists of two 32 bit subwords of  $x_{16}$  and  $x_{15}$ .

## PER ROUND LINEAR FACTOR OF TYPE A

For reasons that will become clear momentarily, we would like to find a matrix  $A$ , and a function  $\psi$  such that

$$A\phi(x) = \psi(Ax). \quad (3)$$

for all  $x$ . Under these conditions, we say we have an  $A$  factor, in honor of  $A$  occurring in equation (3) above. If (1), (2), and (3) hold,

$$Ax_{i+1} + Ax_{i-1} = \psi(Ax_i + Ak_i)$$

yielding

$$y_{i+1} + y_{i-1} = \psi(y_i + l_i) \quad (4)$$

where  $y_i = Ax_i$ ,  $l_i = Ak_i$ . Equation (4) is identical in form to equation (2), so the pairs  $(y_i, l_i)$  form a new cipher system. We call this the "mapped" cipher system.  $y_0, y_1$  form the mapped plaintext,  $y_{15}$  and  $y_{16}$  form the mapped ciphertext and the  $l_i$  are the mapped per round keys.

Let  $KS_i$  be the key schedule matrix for round  $i$ , then the map

$$\mathbf{k} \longmapsto (KS_1(\mathbf{k}), \dots, KS_{16}(\mathbf{k}))$$

has an image (in  $V_{768}$ ) of dimension 56. If the corresponding key schedule for the mapped cipher, given by

$$(A KS_1(\mathbf{k}), \dots, A KS_{16}(\mathbf{k})) = \mathbf{l} = (l_1, \dots, l_{16}),$$

has dimension  $n$  ( $0 < n < 56$ ), we can recover the original key as follows. Search over the mapped key space to find the  $\mathbf{l}$  producing the correct behavior in a transformed plain/ciphertext pair. This costs  $2^n$  time. Then go back to the original cipher, looking for the key  $\mathbf{k}$  in the coset of the null space of  $A$  mapping to  $\mathbf{l}$ . This costs  $2^{56-n}$  time. Total cost:  $2^n + 2^{56-n}$ .

We need some more notation for later, most of the notation concerns projection operators of various sorts to wit:

$$\pi_i(x_1, \dots, x_n) = (0, 0, \dots, x_i, 0, \dots, 0)$$

$$\theta_{j,m} = \pi_j + \dots + \pi_m$$

$$\rho_i = \theta_{6(i-1)+1, 6i}$$

$$\rho'_i = \theta_{4(i-1)+1, 4i}$$

$$V^{(i)} = \rho_i(V_{48})$$

$$V^{(i)} = \rho'_i(V_{32})$$

$$W^{(i)} = E(V^{(i)})$$

$$\phi_x(y) = \phi(x+y) - \phi(x)$$

$N_A$  is the null space of  $A$ .

## LOOKING FOR AN $A$

Now we show that no such non trivial  $A$  exists. The following characterization will facilitate the search for  $A$ . Statement 1 is the one we want for cryptanalysis. Statement 2 is easier to verify; statement 3 is still easier to verify.

**THEOREM 1.** Suppose  $A:V \longrightarrow V$  and  $\phi:V \longrightarrow V$ , with  $A$  linear. The following are equivalent.

1. There is a  $\psi:W \longrightarrow W$  such that  $A\phi(x) = \psi(Ax)$ .
2. If  $Ax = Ay$  then  $A\phi(x) = A\phi(y)$ .
3. For all  $x$  in  $V$ ,  $\phi_x(N_A) \subseteq N_A$ .

**PROOF.**  $3 \implies 2$ : If  $Ax = Ay$ ,  $A(x-y) = 0$  so  $x-y$  is in  $N_A$ . By the conclusion of 3,  $A(\phi(z+(x-y)) - \phi(z)) = 0$  for all  $z$  in  $V$ . Setting  $z = y$  and distributing the  $A$ , we get  $A\phi(x) = A\phi(y)$ .

$2 \implies 3$ : If  $z$  is in  $N_A$ ,  $A(x+z) = Ax$  for any  $x$ ; so, by 2,  $A\phi(x+z) = A\phi(x)$ . Thus,  $A(\phi(x+z) - \phi(x)) = 0$ ; so  $\phi(x+z) - \phi(x)$  is in  $N_A$ .

$1 \implies 2$ :  $A(\phi(x) - \phi(y)) = \psi(Ax) - \psi(Ay) = 0$ , the last equality follows from 1 if  $x = y$ .

$2 \implies 1$ : Define  $\psi(Ax) = A\phi(x)$ . We need only show that the given map is well defined. If  $Ax = Ay$ ,  $A\phi(x) = A\phi(y)$ , so the map is well defined. Note that  $W$  is just the image (in  $V$ ) of  $A$ . This condition insures that the diagram below commutes.



$$\begin{array}{ccc}
 V & \xrightarrow{\phi} & V \\
 A \downarrow & & A \downarrow \\
 W & \xrightarrow{\psi} & W
 \end{array}$$

Commuting diagram for  $2 \Rightarrow 1$

By Theorem 1 (3), we want to look for subspaces  $S$  satisfying the following condition.

CONDITION S.  $\phi_x(S) \subseteq S$  for all  $x$  in  $V$ .

THEOREM 2. Let  $T_i(a) = \text{span}\{S_i(a+b) - S_i(b), \text{ all } b \text{ in } V_6\}$ . If  $i \neq 4$  and  $a \neq 0$  then  $T_i(a)$  is  $V^{(i)}$ . If  $i=4$  and  $a \neq 0$ ,  $T_i(a)$  is one of two 2 dimensional spaces, a 3 dimensional space or the entire 4 dimensional space,  $V^{(i)}$ .

PROOF. A simple computer program was written to verify these.

THEOREM 3. Suppose  $S$  is a subspace satisfying "condition S". Further, suppose there is a  $y$  in  $S$  with  $\rho_i(y) \neq 0$ . If  $i \neq 4$ ,  $W^{(i)} \subseteq S$ ; if  $i=4$ ,  $S$  contains at least a two dimensional subspace of  $W^{(i)}$ .

PROOF. Suppose  $u, v$  are in  $V^{(i)}$ . Then by condition S,

$$u^* = \phi(y+u) - \phi(u) = EP(S_1(\rho_1(u+y)) - S_1(\rho_1(u)), \dots, S_8(\rho_8(u+y)) - S_8(\rho_8(u)))$$

and

$$v^* = \phi(y+v) - \phi(v) = EP(S_1(\rho_1(v+y)) - S_1(\rho_1(v)), \dots, S_8(\rho_8(v+y)) - S_8(\rho_8(v)))$$

are in  $S$ .  $u^* - v^*$  must also be in  $S$  and  $\rho_j(u+y) = \rho_j(v+y) = \rho_j(y)$  if  $j \neq i$ . So,

$$u^* - v^* = EP(0, 0, \dots, 0, S_i(\rho_i(u+y)) - S_i(\rho_i(u)) - S_i(\rho_i(v+y)) + S_i(\rho_i(v)), 0, \dots, 0)$$

is also in  $S$ . Setting

$$T(u, v) = S_i(\rho_i(u+y)) - S_i(\rho_i(u)) - S_i(\rho_i(v+y)) + S_i(\rho_i(v)),$$

theorem 2 tells us that  $\text{span}\{T(u, v): u, v \in V^{(i)}\}$  is all  $V^{(i)}$  if  $i \neq 4$  and is at least a two dimensional subspace of  $V^{(i)}$  if  $i=4$ . Thus, if  $i \neq 4$   $S$  contains  $EP(V^{(i)}) = W^{(i)}$ ; if  $i=4$ ,  $EP(S)$  is (at least) a two dimensional subspace of  $W^{(i)}$ . QED.

REMARK. We say output block  $k$  is affected by input block  $i$  if at least one of the bits of  $V^{(k)}$  is calculated using  $S$  box  $i$ .  $EP$  switches and expands outputs from the  $S$  box calculation so its easy to see that output block  $k$  is affected by input block  $i$  iff  $V^{(k)} \cap W^{(i)} \neq 0$ .

Effect of 6 bit (input, output) blocks on $(x, \phi(x))$		
In block	Out block (round 1)	Out block (round 2)
1	7,4,2,5,6,8	all blocks
2	6,8,3,7,5,1	all blocks
3	5,1,4,6,7,2	all blocks
4	7,2,5,8,3,1	all blocks
5	3,1,2,6,4,8	all blocks
6	4,8,7,1,3,5	all blocks
7	3,5,4,2,8,6	all blocks
8	2,6,3,1,7,4	all blocks

LEMMA. Suppose  $S$  is a subspace satisfying "condition S" and suppose  $i \neq 4$ . If a 6 bit output block  $k$  is affected, during the calculation of  $\phi(x)$ , by a bit from a six bit input block  $i$  and if  $S$  contains a  $y$ , such that  $\rho_i(y) \neq 0$ , then  $W^{(k)} \subseteq S$  provided  $k \neq 4$ . If  $k=4$ , there is at least a two dimensional subspace of  $W^{(k)}$  contained in  $S$ .

PROOF. If output block  $k$  is affected by input block  $i$ ,  $\rho_k(W^{(i)}) \cap V^{(k)} \neq 0$ . Since  $\rho_i(y) \neq 0$ , theorem 3 yields  $W^{(i)} \subseteq S$ ; this, in turn, means there is a  $y$  in  $S$  such that  $\rho_k(y) \neq 0$ . Applying theorem 3 again, we get  $W^{(k)} \subseteq S$ , if  $k \neq 4$ ; if  $k=4$  there is a two dimensional subspace,  $W'$ ,  $W' \subseteq S$ , with  $W' \subseteq W^{(k)}$ . This is exactly what the lemma claims, so we are done. QED.

THEOREM 4. If  $S$  is a subspace satisfying "condition S" and  $S \neq 0$  then  $S = W$ .

PROOF. We prove this by pumping up  $S$  to  $W$ . Suppose  $S \neq 0$ , then there is an  $i$  ( $1 \leq i \leq 8$ ) and a  $y$  in  $S$  with  $\rho_i(y) \neq 0$ .

For the sake of simplicity, let's assume  $i=1$ , so  $\rho_1(y) \neq 0$ . By theorem 1,  $W^{(1)} \subseteq S$ ; by the Lemma,  $W^{(k)} \subseteq S$ , for  $k = 2, 5, 6, 7, 8$  and, in addition, there's at least a 2 dimensional subspace of  $W^{(4)}$  in  $S$ . Now,  $W^{(7)} \subseteq S$  so by reapplying the Lemma, we get  $W^{(k)} \subseteq S$ , for  $k = 2, 3, 5, 6, 8$ . To recap,  $\rho_1(y) \neq 0$  implies that  $W^{(k)} \subseteq S$  for  $k \neq 4$ .

$\sum_{k=1,3,5,6,8} (W^{(k)} \cap V^{(4)}) = V^{(4)}$ . It's easy to see that  $V^{(4)} \subseteq S$  implies  $W^{(4)} \subseteq S$ . Thus  $W^{(k)} \subseteq S$  for  $1 \leq k \leq 8$ , hence  $S = W$ .

For values of  $i$  other than 1 and 4 the argument in the preceding paragraph applies *mutatis mutandis*. If  $\rho_4(y) \neq 0$ , the table and Theorem 3 show that  $W^{(4)} \cap V^{(n)} \neq 0$  for some  $n$  in  $\{1, 2, 3, 5, 7, 8\}$ . Thus, for some  $y$  in  $S$   $\rho_i(y) \neq 0$ , for some  $i \neq 4$  provided only that  $S \neq 0$ . By the above argument,  $S = W$ . QED.

REMARK. The proof of the theorem above basically "reapplies" the mapping  $\phi_x(y)$  for non zero  $y$  in  $S$  until  $\phi_x$  gobbles up  $S$ .

## EXTENSION TO AB FACTORS

We have called the sort of per round linear factor discussed above an *A factor*, in honor of the equation

$$A\phi(x) = \psi(Ax)$$

which holds for all  $x$ . A fancier kind of factor is the *AB factor*, which we now discuss. Here we suppose we are given a pair of linear maps  $A$  and  $B$ , and a possibly non linear function  $\psi$ , so that for all  $x$  both of

$$A\phi(x) = \psi(Bx)$$

and

$$B\phi(x) = \psi(Ax)$$

hold. Clearly an *A factor* is an *AB factor*: just let  $B = A$ .

A non trivial *AB factor* can also be used to solve the DES. One applies  $A$  and  $B$  alternately to the DES rounds. Let

$$T_i = A$$

if  $i$  is even and

$$T_i = B$$

if  $i$  is odd. Then

$$y_i = T_i x_i$$

$$l_i = T_i k_i$$

and

$$y_{i+1} + y_{i-1} = \psi(y_i + l_i)$$

as in equation (4) above. This is a factor cryptosystem of DES type, but it may have a smaller keyspace.

Unfortunately, if an AB factor exists, so does an A factor. This follows from the following fact, whose proof is easy:

THEOREM 5. Let

$$\phi: V \longrightarrow V$$

$$\psi_1: W_1 \longrightarrow W_2$$

$$\psi_2: W_2 \longrightarrow W_1$$

$$T_1: V \longrightarrow W_1$$

$$T_2: V \longrightarrow W_2$$

be maps between vector spaces,  $\phi$ ,  $\psi_1$ , and  $\psi_2$  not necessarily being linear. Suppose, for all  $v$  in  $V$  we have

$$T_2(\phi(v)) = \psi_1(T_1(v))$$

and

$$T_1(\phi(v)) = \psi_2(T_1(v)).$$

Then there is a vector space  $W$  and a linear map  $A: V \longrightarrow W$  and a function  $\psi: W \longrightarrow W$  such that for all  $v$  in  $V$ ,

$$A(\phi(v)) = \psi(A(v)).$$

PROOF. Let  $W = W_1 \oplus W_2$ . Then  $A(v) = (T_1(v), T_2(v))$  and  $\psi(w_1, w_2) = (\psi_1(w_1), \psi_2(w_2))$  satisfy the conclusion of the theorem. QED.

## EXTENSION TO $\alpha\beta\gamma$ FACTORS

Stepping back a moment, we might say that the point of the above attack is to find a per round *linear* relationship among the (*plaintext, ciphertext, key*) triples. If we don't insist that the relationship be linear, a broader attack may hold. For example, consider the following, which we call an  $\alpha\beta\gamma$  factorization.

Let  $\sigma_i$  be a basic enciphering operation (like a round of DES) depending on the key bits  $k_i$ . Then the plaintext  $x_0$  is converted to the ciphertext  $x_n$  by the iteration  $x_{i+1} = \sigma_i(x_i)$ . Suppose we can find  $\alpha$ ,  $\beta$ , and  $\gamma$  with  $\gamma$  linear satisfying

$$\alpha(x) + \beta(\sigma_i(x)) + \gamma(k_i) = 0 \quad (5)$$

and

$$\beta(x) + \alpha(\sigma_i(x)) + \gamma(k_i) = 0. \quad (6)$$

(Equivalently, we might require

$$\alpha(x) + \alpha(\sigma_i(x)) = \beta(\sigma_i(x)) + \beta(x) \quad (7)$$

instead of (6).) Now we can apply (5) to the enciphering equations, yielding (term by term)

$$\alpha(x_i) + \beta(x_{i+1}) = \gamma(k_i)$$

and, on summation,

$$\sum_{i=0}^{n-1} \alpha(x_i) + \sum_{i=0}^{n-1} \beta(x_{i+1}) = \sum_{i=0}^{n-1} \gamma(k_i).$$

Rearranging and using (2) and canceling terms appearing an even number of times, we get

$$\alpha(x_0) + \alpha(x_n) = \sum_{i=0}^{n-1} \gamma(k_i) \quad (7)$$

when  $n$  is even and

$$\beta(x_0) + \alpha(x_n) = \sum_{i=0}^{n-1} \gamma(k_i)$$

when  $n$  is odd. Belaboring the point, we might write

$$\alpha(\text{plaintext}) + \alpha(\text{ciphertext}) = \sum_{i=0}^{n-1} \gamma(k_i) \quad (8a)$$

or

$$\beta(\text{plaintext}) + \alpha(\text{ciphertext}) = \sum_{i=0}^{n-1} \gamma(k_i). \quad (8b)$$

To use such a relation to help find the key, suppose we are trying to find a key in  $V_k$ . Let  $W$  be the  $k-1$  dimensional subspace satisfying (8a) or (8b). Instead of searching all elements in  $V_k$ , restrict the search to elements of  $W$ . This produces a computational savings of  $1/2$ ; if many such relations can be found, determination of the key (even for a large keyspace) would be quick and painless.

Whether  $\alpha, \beta, \gamma$  satisfying (5) and (6) exist is a deep question. Sometimes their existence and discovery are not too difficult. For example, Equation (6) automatically holds if (5) holds and  $\sigma_i$  is an involution. Significantly, it is easy to show that (6) also holds for a round of DES (where  $\sigma_i = \mu \lambda_i$ ) if (5) holds with  $\sigma_i = \lambda_i$  and  $\alpha(\mu(x)) + \beta(\mu(x)) = \alpha(x) + \beta(x)$ . If an S box had a non trivial affine dependence, we could manufacture such functions in the following manner. Suppose we had

$$MS(x) + L(x) = 0$$

for some S box  $S$  with  $M$  and  $L$  matrices of size  $1 \times 4$ . Set  $\alpha = M$  and  $\beta = \gamma = L$ . As a consequence of the above relation, we have

$$\alpha(S(x_i)) + \beta(x_i) = 0$$

or since  $\beta = \gamma$  is linear

$$\alpha(S(x_i)) + \beta(x) = \gamma(k_i)$$

It is easy to see how to modify  $\alpha, \beta$  and  $\gamma$  when we replace  $S$  by the  $\sigma_i$  of DES.

Once again, no S box has this linear property. But this begs the larger question: *Do any such  $\alpha, \beta, \gamma$  exist for DES?* Unfortunately, it can be shown that any  $\alpha\beta\gamma$  factor already takes this form. To see this, it will be convenient to switch notation. Writing a round of DES as

$$(x, y) \mapsto (y, x + f(y + k_i))$$

equation (5) becomes

$$\alpha(x, y) + \beta((y, x + f(y + k_i))) + \gamma(k_i) = 0 \quad (9)$$

$f$ , being the cryptographic function defined in the section on DES notation. Now set  $y = 0$ , and, as before, let  $\phi(x) = Ef(x)$ . Let  $P$  be a quasi inverse of  $E$  on  $V_{32}$ , i.e.,  $PE(x) = x$  for  $x$  in  $V_{32}$ . Now make the following definitions (Caution: the  $f$  below is not the same as the  $f$  in equation (5); also, remember that  $P$  is *not* the  $P$  defined in the section on DES notation.)

$$g(x) = \alpha(Px, 0)$$

$$h(x) = \gamma(x)$$

$$f(x) = \beta(0, Px)$$

Then for all  $x$  and  $y$  in  $W (=V_{48})$ ,

$$g(x) + h(y) = f(x + \phi(y))$$

We now show that the above equation holds only if  $f$  is affine.

THEOREM 6. Suppose  $\phi: W \longrightarrow W$  and that for all  $x, y$  in  $W$

$$f(x + \phi(y)) = g(x) + h(y)$$

where  $h$  is linear and  $g(a) = 0$  for some  $a$  in  $W$ . If  $\text{Image}(\phi)$  is an abelian group then  $f$  is affine on  $\text{Image}(\phi)$ . Since  $f$  is affine and  $h$  is linear,  $g$  is also affine.

PROOF.

$$f(x + \phi(y_1 + y_2)) = g(x) + h(y_1) + h(y_2) = f(x + \phi(y_1)) + h(y_2) = f(x + \phi(y_2)) + h(y_1) \quad (*)$$

Since we are in  $\text{GF}(2)$ ,

$$f(x + \phi(y_1)) = f(x + \phi(y_2)) + h(y_1) + h(y_2)$$

$g(a) = 0$  and  $(*)$  imply

$$f(x + \phi(y_1)) = f(x + \phi(y_2)) + f(a + \phi(y_1)) + f(a + \phi(y_2))$$

Since the image of  $\phi$  is in  $W$  and  $a$  is in  $W$ , we can set  $x = a + \phi(y_2)$  yielding

$$f(\phi(y_1) + \phi(y_2) + a) = f(a + \phi(y_1)) + f(a + \phi(y_2)) + f(a)$$

Finally, since  $\text{Image}(\phi)$  is an abelian group, for all  $u_1, u_2$  in  $\text{Image}(\phi)$ , we can find  $y_1, y_2$  in  $W$  with  $u_1 = \phi(y_1)$ ,  $u_2 = \phi(y_2)$  giving:

$$f(u_1 + u_2 + a) = f(a + u_1) + f(a + u_2) + c.$$

Setting  $l(x) = f(x + a)$ , this becomes

$$l(u_1 + u_2) = l(u_1) + l(u_2) + c$$

as claimed.

THEOREM 7. DES has no per round linear factors of  $\alpha, \beta, \gamma$  type.

PROOF. If DES had a per round factor of  $\alpha, \beta, \gamma$  type, then by theorem 6, the factor functions would express a (non trivial) affine relationship among the input, output, and key bits of a round. Since the outputs of different S boxes are algebraically independent, it suffices to show that no such relationship holds among the four output bits of any of the eight S boxes. Application of the following lemma concludes the proof.

LEMMA. Let  $S_{4(i-1)+j}$  denote the  $j$ 'th bit of S box  $i$ . Then for all  $i$ ,

$$\sum_{j=1}^4 a_j S_{4(i-1)+j}(x_1, \dots, x_6) + \sum_j^6 b_j x_j + d = 0$$

implies that  $a_j = b_k = 0$  for  $j = 1, 2, 3, 4$ ,  $k = 1, 2, \dots, 6$ .

PROOF. Linear algebra applied to the truth tables of all of the output bits of all of the S boxes. QED.

## CONCLUSION

DES seems to have no non trivial linear per round factor structure. It's hard to imagine a non linear per round factor structure that is useful for cryptanalysis. It is barely possible DES has a non trivial global factor structure that induces trivial factor behavior per round but nobody we know has a clue about what that would look like. The conclusion is that DES will not be solvable by factorization.

Nothing in this note says anything about *approximate* factorizations, or factorizations that *usually* hold, nor have we given up on finding non linear per round factors that yield tractable (non linear) constraint equations.

## References

- [1] National Bureau of Standards, Data Encryption Standard. U.S. Department of Commerce, FIPS 46, 15 January 1977.
- [2] MacWilliams, F.J. and N.J.A. Sloane, The Theory of Error Correcting Codes. North-Holland, 1977.



A MESSAGE AUTHENTICATOR ALGORITHM SUITABLE FOR  
A MAINFRAME COMPUTER

Donald Watts Davies  
Independent Consultant,  
Sunbury on Thames,  
Middx, UK.

## INTRODUCTION

Authenticators are widely used to protect payment messages against active attack. They produce a number, sometimes called a 'MAC' which is a function of the whole message and a secret key. the earlier name for them in banking was 'test-key', but this obsolescent term is confusing to cryptographers.

Several algorithms now in use, such as that of S.W.I.F.T. and the Data Seal are not revealed to the public. Authenticators based on the DEA 1 and the DSA algorithms (decimal shift and add - for decimal calculators) are public but neither is well adapted to mainframe computers.

Bankers Automated Clearing Services (BACS) suggested the need for a 'mainframe authenticator' and together, with a colleague, David Clayden, we developed this one, known in banking circles as 'MAA'.

The algorithm attracted the attention of the 'Test Key Working Party' of the CLCB (Committee of the London Clearing Banks) who arranged for independent testing of the algorithm. It is also being considered by an ISO working group.

## DESCRIPTION

The definition of the algorithm is contained in an NPL Report DITC 17/83 dated February 1983 with the same title as this paper, by D. W. Davies and D. O. Clayden. All I can do here is to sketch out its structure. Serious Study requires a copy of the definition. NPL is the UK National Physical Laboratory at Teddington, Middlesex, TW11 0LW, UK.

The key has two numbers, J and K, each of 32 bits. All words used in the algorithm are 32 bits long. When a new key is installed, a key calculation called the 'Prelude' produces 6 numbers  $X_0, Y_0, V_0, W, S, T$  which are used in the rest of the algorithm. The choice of J, K is unrestricted.

Multiplication is the principal tool of this algorithm and is used in two varieties, modulo  $2^{32} - 1$  and modulo  $2^{32} - 2$ . The prelude is mainly the following calculation:-

$$\begin{aligned}
 X_0 &= J^{(4)}(1) \text{ XOR } J^{(4)}(2) \\
 Y_0 &= \left[ K^{(5)}(1) \text{ XOR } K^{(5)}(2) \right] (1 + P)^2 \\
 V_0 &= J^{(6)}(1) \text{ XOR } J^{(6)}(2) \\
 W &= K^{(7)}(1) \text{ XOR } K^{(7)}(2) \\
 S &= J^{(8)}(1) \text{ XOR } J^{(8)}(2) \\
 T &= K^{(9)}(1) \text{ XOR } K^{(9)}(2)
 \end{aligned}$$

Where 1 and 2 refer to the two moduli  $2^{32} - 1$  and  $2^{32} - 2$  respectively and XOR is bit-wise on a 32 bit word.

The eight bytes J,K are first treated by a procedure to replace any byte which is 0000,0000 or 1111,1111. A resultant number P records the changes made and its use in calculating Yo avoids reducing the key space. The pairs  $X_0, Y_0; V_0, W_0$  and S,T are similarly treated to remove runs of zeros or ones before they are used in the body of the algorithm.

The main part of the calculation (we considered calling it the Fugue) takes in the message in blocks  $M_i$  of size 4 bytes and, for each one repeats the steps in Figure 1. The variables  $X, Y, V$  are initialised to  $X_0, Y_0$  and  $V_0$  respectively. For each block,  $V$  is cyclic shifted left one bit and XORed with  $W$  to produce  $E$ . The + operations are modulo  $2^{32}$ . The constants A, B, C, D are used in the logical operations to set 8 bits of each numbers (F and G) to fixed values. The aim is to avoid bytes of all zeros or all ones in the multipliers F and G, as well as to introduce non-linearity. The two multiplications with different moduli complete the round.

The authenticator value to be produced at the end of the calculation is simply  $Z = X \text{ XOR } Y$ , but after the last message block has been used, the numbers S and T are used as message blocks for two rounds (as if appended to the message) before the final XOR operation. This last part, producing Z, is called the Coda.

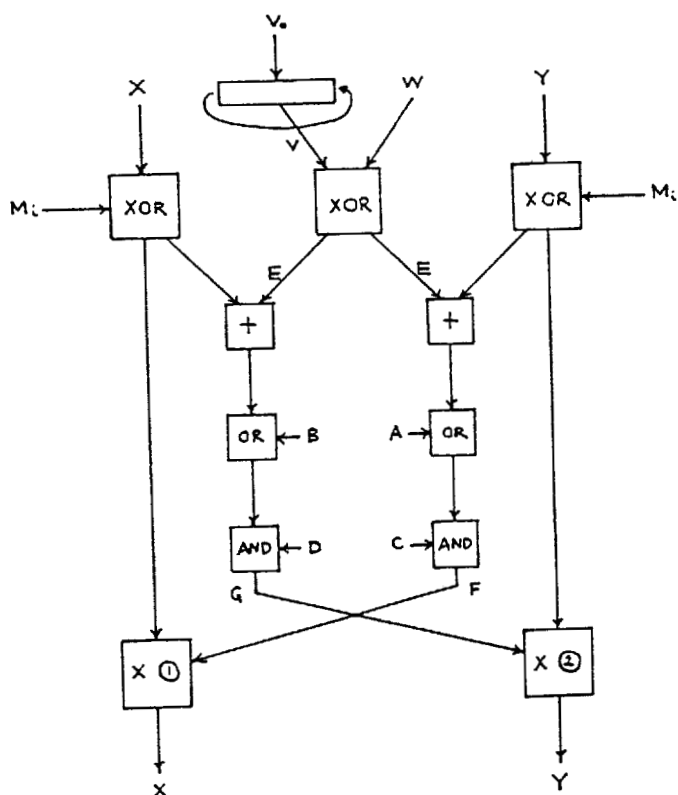


FIGURE 1

## PERFORMANCE

Since this algorithm is designed for a 32 computer containing a multiplier, the performance figure, for a typical IBM configuration would be of interest. But in the time that has passed since the report was published, no such measurements of performance have been reported to us. An assembly language program for a microcomputer (2 MHz 6502 = BBC Micro) takes 47 ms for the prelude and coda and 5.92 ms for each block of message (675 byte/s or 5405 bit/s). Since this uses a programmed multiplication it is not the way that MAA was designed to be used.

## TESTING

We have no positive reason for confidence in the security of the algorithm but at each stage of testing we tried all the input/output dependancies and statistical distributions we could think of. We also used a zero message and some constant messages (such as all ones) and looked for loops.

Most of the testing was done with altered versions of the algorithm deliberately weakened in someway. For example, we demanded in most cases that both the X and the Y values should show good statistical properties (and confirmed the results with Z). We reduced the number of fixed bits in A, B, C, D and removed E or S and T, though not all these at the same time. For sensitivity to key changes we varied separately the six outputs of the prelude, before testing with the prelude in place.

At several stages of development, problems were found and fixed, but we found the fixes had to be carefully thought through to avoid bringing back old problems. When all our weakened tests were passed we tested it again in its complete form.

## PROBLEMS

Two problems have been pointed out. If  $X$  becomes zero and  $M_i$  remains zero then  $X$  remains zero. If you know  $X$  you could make  $M_i = X$  and engineer this zero value. If both  $X$  and  $Y$  becomes zero and  $M_i$  remains zero then  $X$  and  $Y$  remain zero. In this last case any set of consecutive zero message blocks can be inserted without changing the value of the authenticator. This is indeed a flaw but can anyone suggest how an opponent would use it, not knowing when  $X = Y = 0$ , a very rare event?

The second problem was posed by H. Block of SAK Data AB in 'File Authentication - A rule for constructing algorithms', at Eurocrypt 84. If all the  $M_i$  are fixed, each round of the main loop maps  $(X,Y)$  into  $(X,Y)$  with a mapping that is injective. For an approximation, assume that these are random mappings. Now imagine that 3-4 early blocks in a very long sequence are varied so that all  $2^{64}$  states of  $X,Y$  are attained at some point in the sequence of rounds. With constant  $M_i$  values thereafter, each mapping reduces the number of attainable states. When it falls below about  $2^{34}$ , there is a significant risk that values of  $Z$  will be missing from the set of  $2^{32}$ . Eventually, the 'memory' of these early changes of  $M_i$  will be lost. Block concludes that injective functions should never be used. He thinks that the problem may be worse than we see when random mappings are assumed.

During the testing of the algorithm with 'toy' examples this effect was detected and (though with only a few cases to estimate from) its magnitude agreed with the theoretical value for random mappings, so we are content to rely on that theory. If we used the argument that 'it might be much worse' this would disqualify all but provably secure algorithms, of which there is a shortage.

#### ANALYSIS OF THE 'LOSS OF MEMORY' PROBLEM

Suppose that the number of states is  $N$  and that a set  $X_i$  of these is mapped by a random mapping into the same domain, giving  $X_{i+1}$  distinct states. Then approximately (Poisson distribution)

$$\frac{X_{i+1}}{N} = 1 - \exp(-X_i/N)$$

If the sequence is evaluated, starting at  $X_0/N = 1$ , it follows, to a close approximation:

$$\frac{X_i}{N} = 2 / \{i + 1/3 \ln(i) + 9/5\}$$

In the example of the MAA,  $N = 2^{64}$  and  $x_i = 2^{32}$  is reached approximately when  $i = 2^{33}$ . With, say,  $10^9$  blocks of data in the message ( $4 \times 10^9$  bytes), there should be no perceptible effect. In fact, to measure the effect would require a sample of much more than  $2^{33}$  blocks.

We have suggested an arbitrary, but very safe upper limit of  $10^6$  blocks for any one message. Other considerations (error control and recovery) usually set a lower limit than this.

Acknowledgement is made to the National Physical Laboratory for supporting this work and to Open Computer Security for help with the presentation of this paper at Crypto '84.



# KEY MANAGEMENT FOR SECURE ELECTRONIC FUNDS TRANSFER IN A RETAIL ENVIRONMENT

Henry Beker and Michael Walker

Racal Research Limited,  
Worton Drive,  
Worton Grange Industrial Estate,  
Reading, Berkshire, England.

## 1. INTRODUCTION

In this paper we consider a system whose function is to enable users to pay for goods or services by direct electronic transfer of funds. The system consists of terminals, located at retail outlets, which can communicate with acquirers representing various financial institutions.

Each user of the system has a plastic card and a personal identification number (PIN) issued by a financial institution represented in the system. Affixed to the plastic card is a magnetic stripe, which bears the card holder's personal account number as well as other data such as the expiry date of the card. The PIN is typically four decimal digits long, and is effectively the card holder's electronic signature. He is expected to treat it as such, and refrain from divulging it to unauthorised third parties. Of course, the card holder in turn expects that his PIN will be adequately protected by any authorised body which has knowledge of it.

Throughout this paper we make the simplifying assumption that an acquirer holds a complete data base for every card issued by the financial institutions it represents. In particular, an acquirer has a record of corresponding PINs and card data for all of those cards it is authorised to handle. In addition, we assume that the acquirer is in a position to endorse every transaction made with these cards.

When a card holder wishes to make payment for a purchase from the retailer, the plastic card is presented, and the data encoded on the magnetic stripe is read by the terminal. This gives the terminal data related to the card holder, and identifies the acquirer for the particular transaction. The card holder separately enters his PIN. The retailer enters the details of the purchase, and the terminal then communicates with the acquirer, whose function is to ratify the transaction. This entails checking that the card is valid, that the account contains sufficient funds for the purchase, and that the PIN and card do in fact correspond. It is in this sense that the PIN acts as an electronic signature to authenticate the card holder to the acquirer.

Having completed these tasks, the acquirer informs the terminal that the purchase can proceed (or otherwise), and then arranges for the appropriate funds to be credited to the retailer from the card holder's account.

Clearly, it is essential that the PIN should be kept secret, that the transaction messages should be protected against corruption or intentional change during transmission, and that all parties should authenticate each other. These security requirements can all be satisfied by using cryptographic functions based on block ciphers such as the Data Encryption Algorithm [3]. Indeed, techniques to achieve this are published in a number of papers, notably [4] and [5], which describe PIN encryption and message protection respectively. The main problem is not the performance of the security functions themselves, but rather management of the enciphering keys.

To understand some of the problems involved with key management, let us reconsider our entire system. Clearly, security will certainly be enhanced if all encryption and authentication takes place on an end-to-end basis without additional parties becoming involved. This also has the added advantage that it makes the system network independent, which means that in theory any transmission medium can be used, and in practise it is possible to use an alternative media should the primary one fail. Now our system may well have hundreds of thousands of terminals communicating with a hundred or so acquirers. Thus the problem of distributing the multitude of cryptographic keys is paramount. Consequently, if end-to-end security is to be used, it is obviously desirable that the system is equipped with a procedure for

automatically updating the keys. Of course when attempting to design such a procedure, one must bear in mind that it should not be possible for one acquirer to inadvertently compromise the keys of another. Taking this consideration a little further, it is clearly desirable that disclosure of a key should compromise at most one transaction. A system with this feature offers scant reward for anyone who manages to gain knowledge of a particular key. Finally, it must be stressed that our terminals are at most tamper-resistant and certainly not tamper-proof. In particular, a key housed for any length of time in a terminal cannot really be considered safe enough to be used to encipher a PIN. Once again, this points to the desirability of a system which automatically generates a fresh key for each transaction.

The fresh key per transaction, or transaction key, approach was introduced in [1] as a way of overcoming some of these key management problems. The present article expands on the ideas of that paper, and develops a protocol for key management in the electronic transfer of funds system defined above. It should be stressed that the techniques are applicable to more general systems than the one chosen here. Our choice was made on the basis of requiring a system which reflected many of the central problems of key management, without being so complex that it overshadowed the salient features of the scheme. For a discussion in a wider context the reader is referred to [2]. For simplicity of description, the protocols are defined for a basic request-response message flow between a terminal and an acquirer. The scheme is such that confirmation of (non-) completion at a terminal of a particular transaction is automatically conveyed by the next request from that terminal to the acquirer. This does not mean that the system will not support a confirmation message within a transaction. Indeed, the protocols are readily adapted to accommodate a far more comprehensive dialogue than the one described here.

Throughout the paper, we shall base all cryptographic functions needed to describe our key management protocols on the Data Encryption Algorithm (DEA). Of course, this is just a convenience, and it is not necessary to appreciate how the algorithm works. All that is required is to know that it transforms 64 bit blocks of clear text to 64 bit blocks of cipher text under control of an enciphering key. The key is also 64 bits in length, but only 56 of the bits are actually used by the enciphering algorithm. We shall denote the clear text input to the

algorithm by DATA, the resulting cipher text block by CIPHER and the controlling key by KEY. Thus, for our purposes, the enciphering algorithm E is described by

$$E(\text{KEY}) : \text{DATA} \longrightarrow \text{CIPHER}$$

Having established this notation, we return to our system and describe how the transaction keys are generated.

## 2. TRANSACTION KEYS

We make use of the well known concept of including some card holder dependent data on the magnetic stripe of the card, and refer to this data as the card key. This data is read by the terminal but is not itself transmitted. In addition, the terminals contain a key register for each acquirer with whom it is authorised to communicate.

Let us assume that our system is up and running, and that a card holder, wishing to pay for a purchase, presents his plastic card to the terminal. The terminal then reads the card key and the card holder's personal account number (PAN), and identifies the acquirer for the particular card. The transaction key is then generated at the terminal as a one-way function of the card key and the value in the key register for the particular acquirer.

Assuming the one-way function is to be based upon the DEA, then the transaction key might be generated as follows:

$$\begin{array}{lll} \text{DATA} & \longleftarrow & \text{card key} \\ \text{KEY} & \longleftarrow & \text{key register value} \\ \text{transaction key} & \longleftarrow & \text{CIPHER} \oplus \text{DATA} \end{array}$$

Of course, the terminal must provide the acquirer with sufficient information to generate the transaction key at his end. To accomplish this, the request message from the terminal includes in clear text the card holder's PAN and the terminal's identification. The acquirer maintains a key register for each terminal in the system, and the value in the register for a particular terminal agrees with the value held by that terminal in its key register for the acquirer. Since by

hypothesis the acquirer holds a complete data base for every card it is authorised to handle, the acquirer has a list of corresponding card keys and PANs. Thus the acquirer is able to construct the transaction key by identifying the card key from the PAN and the key register value from the terminal identification.

We have defined the transaction key as a function of two independent variables, the card key and the key register value, which is generated by both parties precisely when it is needed. In general, it is not possible to predict its value in advance because of the element of randomness provided by the card key. Naturally, we must update the key register value at the end of the transaction, and we also wish to make this updating a random process. As we shall see, this second element of randomness is provided by the unpredictability of the messages exchanged during the transaction.

### 3. THE REQUEST MESSAGE

Once the terminal has read the card and constructed the transaction key, details of the transaction are entered and a request message is compiled. This message must include the card holder's PAN and the terminal identification, both in clear text. In our system it will also include the card holder's PIN (or PIN offset), which should be separately entered and enciphered under the transaction key before being inserted in the message. It may also include other cipher blocks as well.

Having compiled the request message, the terminal must now add an ingredient which protects it against change. In addition, the terminal must have a procedure to authenticate the acquirer, and a way of checking that the acquirer's response does indeed pertain to the particular request. These three authentication checks are achieved by generating a message authentication block (MAB) under control of the transaction key. Generation of this block follows the procedure described in [5] for constructing a message authentication code (MAC). The message is divided into  $n$  blocks

$$B_1, B_2, \dots, B_n$$

each 64 bits in length. The DEA is then used to process these blocks sequentially under control of the transaction key. More precisely

KEY  $\longleftarrow$  transaction key

and n data blocks

DATA1, DATA2, ....., DATAn

are sequentially processed under KEY to produce cipher blocks

CIPHER1, CIPHER2, ....., CIPHERn

where

DATA1 = B<sub>1</sub>

and

DATAj = CIPHER(j-1)  $\oplus$  B<sub>j</sub>

for j = 2, 3, ....., n. The request MAB is defined to be the final cipher block CIPHERn.

The left hand half of the MAB forms the request MAC. This is inserted into the message before transmission, and allows the acquirer to verify that the message has not been changed during transmission. The remaining 32 bits of the MAB form the request residue. This is retained by the terminal, and used later to authenticate the acquirer as the originator of the response message, and to confirm that the response corresponds to the request. It is also used in the key register updating procedure.

When the acquirer receives the request message, it generates the transaction key, removes the request MAC from the message, and then uses the algorithm described above to generate a MAB for the truncated message. The left hand half of the MAB is then compared with the request MAC retained from the received message in order to confirm the message integrity. If the two do in fact agree, then this also authenticates the terminal to the acquirer because their key register values must be identical. Once the request message has been

authenticated, the right hand half of the MAB is retained by the acquirer as the request residue.

#### 4. THE RESPONSE MESSAGE

When the acquirer has finished checking the transaction details, he prepares a response message, and generates a MAB under control of the transaction key. The MAB generation procedure for the response message is slightly different from that described for the request message. The acquirer first adds the request residue to the beginning of the response, and then constructs a MAB for the extended message. The rationale for constructing the response MAB on the extended message is to tie the response to the request which prompted it. This enables the terminal to authenticate the acquirer as the originator of the response, as well as to confirm that the response does indeed correspond to the request.

The left hand half of the response MAB is the response MAC. This is inserted into the response message after the request residue has been removed, and is used by the terminal to authenticate the message. The other half of the response MAB is called the response residue. This is retained by the acquirer, along with the request residue, to be used to update the key register.

When the terminal receives the response message, it removes the response MAC, adds the request residue to the beginning of the message, and then generates a MAB for the extended message. The left hand half of the MAB is then compared with the request MAC retained from the incoming response message. If the two agree, then the terminal has authenticated both the message and the acquirer and confirmed that the received response corresponds to the original request message. Once authentication has been completed, the terminal retains the right hand half of the MAB as the response residue, and completes its end of the transaction.

## 5. KEY REGISTER UPDATING

After the acquirer has transmitted the response message, the transaction key is destroyed, the current key register value is stored, and the key register is updated. The reason for retaining a copy of the key register value will be explained later. First, we describe the key register updating procedure.

The new value of the key register is a one-way function of the current value, the request residue and the response residue. If we use the same one-way function as we used to generate the transaction key, then the updating procedure may be defined by:

DATA	←	(request residue, response residue)
KEY	←	key register value
key register value	←	CIPHER $\oplus$ DATA,

where the input to DATA is a concatenation of the request and response residues. Thus the new key register value depends upon the old value and the message residues. Since the message residues depend upon the messages exchanged and the transaction key, which itself depends upon the card key, the new key register value is in general quite unpredictable.

The terminal destroys the transaction key and updates its own key register value after it has authenticated the response message; the updating procedure being identical to that of the acquirer. Of course, if the response message fails to reach the terminal or fails to be authenticated, then the terminal's key register value remains unaltered, although the acquirer's has already been updated. It is precisely for this reason that the acquirer retains a copy of the previous value in its key register, for this enables the system to recover from the situation.

Suppose then that the acquirer has updated its key register, but the terminal has failed to do so. On receipt of the next request from the terminal, the acquirer constructs a new transaction key, and then attempts to authenticate the message. Naturally, this will fail because the terminal is still using the old key register value. The



acquirer then replaces its key register value by the old value it has retained, and generates a transaction key based on this value. If authentication is now successful, then the acquirer recognises that the previous transaction did not complete at the terminal, and can take the necessary action. Moreover, the terminal's and acquirer's key registers now agree, so that synchronisation is recovered and the current transaction can proceed as normal.

The above discussion highlights one other feature of the system. Completion of a transaction at a particular terminal is confirmed to the acquirer by successful authentication of the next request received from that terminal.

## 6. CONCLUSION

There are a number of points to note regarding the key management scheme described in this paper. First, the transaction key is end-to-end and unique to the particular transaction. Even if an unauthorised person gained knowledge of a transaction key and the card key of the next card presented at the terminal for use with the same acquirer, this would not be sufficient to deduce the next transaction key. Similarly, it is not possible to deduce anything about the previous transaction with that acquirer. Thus the rewards for breaking a single key are indeed small. Secondly, key management is automatic, and a transaction key is unpredictable (because it depends on a card key and the value in a key register, and this value depends upon the previous value, the previous card key and the messages exchanged during the previous transaction). Thirdly, confirmation that a transaction completed at a terminal is inherent in the next communication between the terminal and acquirer. Fourthly, it should be noted that a log-in is not required, and the system does not need to be shut down for the purpose of distributing new keys. Fifthly, even if someone breaks into a tamper-resistant terminal and obtains the key register values for some acquirers, the information is useless to them just as soon as bona-fide cards are presented at the terminal for use with these acquirers. This removes some of the need for high physical security of the terminals.

The reader will probably have noted that we have described protocols for a system which is already operational, but have not

mentioned how the system is initialised. This could be handled in several ways, and we shall make three suggestions. First, it is not inconceivable that a public key cryptosystem could be used to provide initial values for all key registers. Alternatively, each acquirer could insist that when a terminal is installed a test transmission using a test-card should precede all other transactions. This same test-card might also be used to re-initialise in the event of a catastrophic failure. A third possibility is that an acquirer might well simply choose to ignore the problem, bearing in mind that once a bona fide card is presented the terminal - acquirer link attains full security.

Finally, we mention once again that the key management scheme outlined in this paper is applicable to more general systems than the one we have described. The techniques can be adapted to cover the situation where the acquirer does not hold a complete data base for each card it is authorised to handle, and to provide for a more extensive dialogue between terminal and acquirer.

#### REFERENCES:

1. H.J. Beker, J.M.K. Friend, P.W. Halliden; Simplifying key management in electronic fund transfer point of sale systems, *Electronics Letters* Vol.19 No.12 (1983), 442-444.
2. H.J. Beker; Secure electronic funds transfer in a retail environment; *Networks 84, Proceedings of the European Computer Communications Conference* (1984), 263-270.
3. American National Standard; "Data Encryption Algorithm", ANSI X3.92 (1981).
4. American National Standard; "Pin Management and Security", ANSI X9.8 (1982).
5. American National Standard; "Financial Institution Message Authentication", ANSI X9.9 (1982).

# AUTHENTICATION THEORY/CODING THEORY\*

Gustavus J. Simmons

Sandia National Laboratories  
Albuquerque, New Mexico 87185

## ABSTRACT

We consider a communications scenario in which a transmitter attempts to inform a remote receiver of the state of a source by sending messages through an imperfect communications channel. There are two fundamentally different ways in which the receiver can end up being misinformed. The channel may be noisy so that symbols in the transmitted message can be received in error, or the channel may be under the control of an opponent who can either deliberately modify legitimate messages or else introduce fraudulent ones to deceive the receiver, i.e., what Wyner has called an "active wiretapper" [1]. The device by which the receiver improves his chances of detecting error (deception) is the same in either case: the deliberate introduction of redundant information into the transmitted message. The way in which this redundant information is introduced and used, though, is diametrically opposite in the two cases.

For a statistically described noisy channel, coding theory is concerned with schemes (codes) that introduce redundancy in such a way that the most likely alterations to the encoded messages are in some sense close to the code they derive from. The receiver can then use a maximum likelihood detector to decide which (acceptable) message he should infer as having been transmitted from the (possibly altered) code that was received. In other words, the object in coding theory is to cluster the most likely alterations of an acceptable code as closely as possible (in an appropriate metric) to the code itself, and disjoint from the corresponding clusters about other acceptable codes.

\* This work performed at Sandia National Laboratories supported by the U. S. Department of Energy under Contract No. DE-AC04-76DP00789.

In [1,2] the present author showed that the problem of detecting either the deliberate modification of legitimate messages or the introduction of fraudulent messages; i.e., of transmitter and digital message authentication, could be modeled in complete generality by replacing the classical noisy communications channel of coding theory with a game-theoretic noiseless channel in which an intelligent opponent, who knows the system and can observe the channel, plays so as to optimize his chances of deceiving the receiver. To provide some degree of immunity to deception (of the receiver), the transmitter also introduces redundancy in this case, but does so in such a way that, for any message the transmitter may send, the altered messages that the opponent would introduce using his optimal strategy are spread randomly, i.e., as uniformly as possible (again with respect to an appropriate metric) over the set of possible messages,  $\mathcal{M}$ . Authentication theory is concerned with devising and analyzing schemes (codes) to achieve this "spreading." It is in this sense that coding theory and authentication theory are dual theories: one is concerned with clustering the most likely alterations as closely about the original code as possible and the other with spreading the optimal (to the opponent) alterations as uniformly as possible over  $\mathcal{M}$ .

The probability that the receiver will be deceived by the opponent,  $P_d$ , can be bounded below by any of several expressions involving the entropy of the source  $H(S)$ , of the channel  $H(M)$ , of the encoding rules used by the transmitter to assign messages to states of the source  $H(E)$ , etc. For example:

$$(1) \quad \log P_d \geq H(MES) - H(E) - H(M)$$

The authentication system is said to be perfect if equality holds in (1), since in this case all of the information capacity of a transmitted message is used to either inform the receiver as to the state of the source or else to confound the opponent. In a sense, inequality (1) defines an authentication channel bound similar to the communication channel bounds of coding theory. Constructions for perfect authentication systems are consequently of great interest since they fully realize the capacity of the authentication channel. In the paper given at Crypto 84 we analyzed several infinite families of perfect systems and also extended the channel bounds to include cases in which the opponent knew the state of the source. Here we have the more modest goal of rigorously deriving the channel bound (1) and then using this result to derive a family of related bounds.

## FUNDAMENTALS

In authentication, there are three participants: a transmitter who observes an information source  $\mathcal{S}$  and wishes to communicate these observations to a remotely located receiver over a publicly exposed, noiseless, communications channel and a

receiver who wishes to not only learn what the transmitter has observed but also to assure himself that the communications (messages) that he receives actually came from the transmitter and that no alterations have been made in transit to the messages sent by the transmitter. The third participant, the opponent, wishes to deceive the receiver into accepting a message that will misinform him as to the state of the source. He can achieve this end in either of two ways: by impersonating the transmitter and sending a fraudulent message to the receiver when in fact none has been sent by the transmitter, or else by waiting and intercepting a message sent by the transmitter and substituting some other message. There are two possibilities to be considered; the opponent may either know or not know the state of the source; he does however know the message sent by the transmitter. Using this information, in either case, he can choose some other message to forward to the receiver. The opponent "wins" if the receiver accepts the fraudulent message in any of these situations as being a genuine (authentic) communication from the transmitter, and thereby ends up being misinformed about the state of the source. We have defined the authentication problem in its narrowest sense here; however, the model can be easily extended to include cases in which the source can be influenced (controlled) by either the transmitter or the opponent or in which the opponent's objectives are more restricted -- i.e., he may wish to deceive the receiver into believing the source is in some particular state(s) not merely an arbitrary deception of the receiver. It is beyond the scope of this paper to treat these other authentication concerns, however, it is essential that the reader appreciate the precise constraints on the model of authentication used here. One of the simplifying assumptions made is that the transmitter and receiver act with common purpose, i.e., that they trust each other completely and that neither acts (either alone or in collaboration with an opponent) to deceive the other. In general, especially in commercial applications, this is an unrealistic assumption, since in practice the transmitter may wish to disavow messages (authentic) that he originated, or the receiver may wish to falsely attribute messages to the transmitter -- or even disclaim having received an authentic message actually sent by the transmitter (and received by him). These questions get into areas of digital signatures, notarization, dating, certification (in the sense of certified mail), etc., which, while closely related to authentication, are primarily questions of systems protocol in which message authentication plays an essential part. We also assume (here) that only the receiver need be convinced of the authenticity of a message -- as opposed to either the transmitter or receiver having to convince a third party (arbiter). In addition, as already mentioned we assume that all successful deceptions of the receiver are of equal value to the opponent, i.e., that his objective is purely to misinform the receiver about the state of the source -- not to cause him to conclude that it is in any particular state. Even though the most interesting applications of digital message authentication made thus far [3,4] have been in situations in which the opponent knew the state of the source (message authentication without

secrecy) we shall mostly be concerned with message authentication in situations in which the opponent is ignorant of the information being communicated to the receiver by the transmitter. Subject to these constraints, we now describe the general authentication system model.

There is a source (set)  $\mathcal{S}$  with a probability distribution  $S$  on its elements for which the binary entropy is  $H(S)$ .  $H(S)$  is the average amount of information about the source communicated to the receiver by the transmitter in each message. There is also a message space  $\mathcal{M}$  consisting of all of the possible messages that the transmitter can send to the receiver. Since an unstated assumption is that the transmitter can communicate to the receiver any observation he makes of the source,  $|\mathcal{M}| \geq |\mathcal{S}|$  where  $|\mathcal{S}|$  is interpreted to be the cardinality of states of  $\mathcal{S}$  that have nonzero probability of occurrence. It should be obvious that authentication depends on the set of messages that the receiver may receive being partitioned into two nonempty parts: a collection of messages that the receiver will accept as authentic and another collection that he will reject as inauthentic. If  $|\mathcal{M}| = |\mathcal{S}|$ , all messages would have to be acceptable to the receiver, hence no authentication would be possible in this case. Therefore,  $|\mathcal{M}| > |\mathcal{S}|$  and as we shall see later the even stronger inequality  $H(M) > H(S)$  holds as well. Figure 1 schematically shows the essential features of what has been described thus far

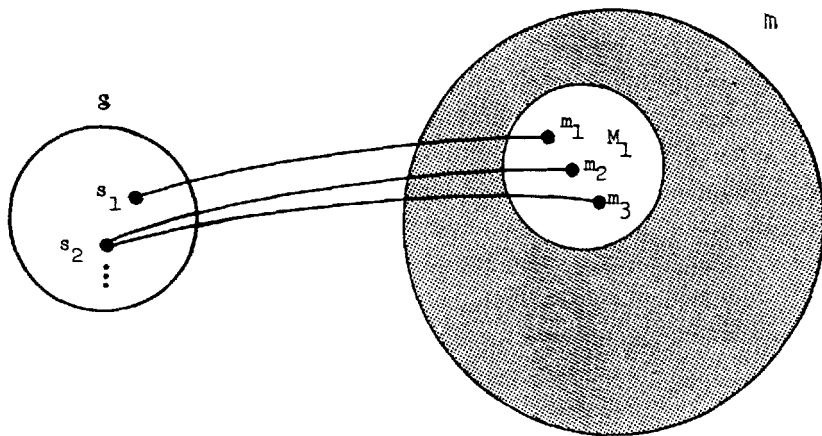


Figure 1.

Any message in the shaded region of  $\mathcal{M}$  would be rejected by the receiver, while any message in the set  $M_1$  would be accepted as authentic. Figure 1 also illustrates that it is possible for the opponent to fail to deceive the receiver, even though he succeeds in getting him to accept a message that was not sent by the transmitter. Assume that the state of the source is  $s_2$  and that the transmitter chooses to encode this information by sending message  $m_2$  to the receiver. If the opponent -- not

knowing the information shown in Figure 1 of course -- intercepts the message  $m_2$  and replaces it with  $m_3$ , the receiver would accept  $m_3$  as being authentic since it is one of the messages that the transmitter might have sent, even though it was not the message actually sent in this case. However the receiver would interpret  $m_3$  to mean that the source state was  $s_2$  -- as observed by the transmitter. The opponent would lose in this case, in spite of the fact that he succeeded in having the receiver accept a fraudulent message, since the receiver is not misinformed as to the state of the source.

There is a well known precept in cryptography, known as Kerckhoff's principle, that the opponent knows the system, i.e., the information contained in Figure 1. It is equally reasonable to assume the same for authentication. Consequently there would be no authentication possible for the receiver using the scheme shown in Figure 1 alone. What is done instead is to have many such encoding rules in an authentication system -- all of which are known to the opponent -- with the choice of the particular encoding rule in use being known only to the transmitter and receiver, similar in many respects to the "key" known only to the transmitter and receiver in a cryptosystem. Figure 2 suggests the general scheme:

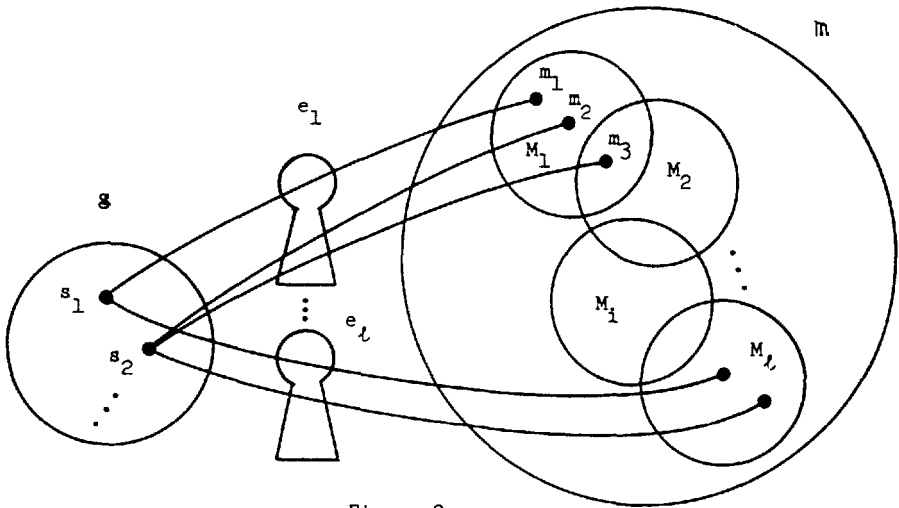


Figure 2.

Each encoding rule,  $e_i$ , determines a proper subset  $M_i$  of  $M$ ,  $|M_i| \geq |S|$ , and a mapping -- perhaps one to many -- of  $S$  onto  $M_i$ . The inverse mapping  $D$  is a well defined function, i.e., for any  $e \in \mathcal{E}$  and  $m \in M_i$ , the function  $D(e, m)$  defines a unique state in  $S \cup \phi$ , where  $\phi$  is the null set.

Even this very intuitive description of authentication should make clear the reason for describing authentication as a problem in "spreading" messages in  $M$ . If  $m_1$  is an acceptable message only in set  $M_1$ , then the opponent, knowing the system, would be able to conclude that  $e_1$  was the coding rule being used if he saw  $m_1$  in the

channel and would then be able to substitute another message with certainty of deceiving the receiver. To avoid this it is necessary that each message occur in sufficiently many authenticating sets to (ideally) leave the opponent no more able to "guess" at an acceptable message after he has observed what the transmitter sent than he could have before the observation. This ideal can be achieved in infinitely many perfect authentication systems [5,6].

#### THE "GAME" MODEL OF AUTHENTICATION

A concise representation of the authentication system depicted in Figure 2 is possible in the form of an  $|\mathcal{E}| \times |\mathcal{M}|$  matrix,  $A$ , where  $\mathcal{E}$  is the set of encoding rules. The rows of  $A$  are indexed by encoding rules and the columns by messages. The entry in  $a(e_i, m_j)$  is the element of  $\mathcal{S}$  encoded by rule  $e_i$  into message  $m_j$  if such a source mapping exists under  $e_i$  and 0 otherwise. Every element of  $\mathcal{S}$  appears in each row of  $A$  at least once and perhaps several times. We define an authentication system to be the triple  $(\mathcal{S}, S, A)$ . Earlier comments imply that each row and column contains at least one 0 entry. We now define another  $|\mathcal{E}| \times |\mathcal{M}|$  matrix  $X$ , in which

$$x(e_i, m_j) = \begin{cases} 1 & \text{if } a(e_i, m_j) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}.$$

For example, for  $|\mathcal{S}| = 2$ ,  $|\mathcal{M}| = 4$ , the "best" authentication system possible has:

$$A = \begin{bmatrix} s_1 & s_2 & 0 & 0 \\ s_1 & 0^2 & s_2 & 0 \\ 0^1 & s_2 & 0^2 & s_1 \\ 0 & 0^2 & s_2 & s_1 \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

It is now easy to see the relationship of the impersonation "game" to the matrix  $X$ . If  $m_j$  is an acceptable (authentic) message to the receiver when encoding rule  $e_i$  has been agreed to by the transmitter and receiver then  $x(e_i, m_j) = 1$  and the opponent has a probability of success of  $p = 1$  if he communicates  $m_j$  to the receiver. Conversely, whenever  $x(e_i, m_j) = 0$  he is certain the message will be rejected. It is certainly plausible -- and in fact rigorously true -- that the opponents probability of success in impersonating the transmitter is the value,  $v_I$ , of the zero sum game whose payoff matrix is  $X$ . It is possible to define a companion payoff matrix  $Y$  for the substitution game, although it is considerably more complex. The value of this game,  $v_S$ , is the probability that the opponent will be successful in deceiving the receiver through intercepting a message sent by the transmitter and substituting one of his own devising. Given an authentication system the transmitter/receiver have



the freedom to choose among the encoding rules and if some state(s) of the source can be encoded into more than one message under some of the encoding rules, a choice of which messages to use, i.e., a splitting strategy. The opponent on the other hand can choose between impersonation and substitution with whatever probability distribution he wishes and then choose according to his optimal strategy which fraudulent message he will communicate to the receiver, either with no conditioning if he is impersonating the transmitter or else conditioned on the message he observed if he is substituting messages. Not surprisingly there exist authentication systems in which the optimal strategy for the opponent is either pure impersonation, pure substitution, immaterial mixes of the two, or most interesting -- essential mixing of both as well as examples in which splitting is essential in the transmitter/receiver's optimal strategies. The point of these remarks is that we have shown in earlier papers that an opponent's overall probability of success in deceiving the receiver,  $P_d$ , is simply the value of the game whose payoff matrix is the concatenation of  $X$  and  $Y$ , and hence that

$$(2) \quad P_d = v_G \geq \max(v_I, v_S)$$

It is not germane to this paper to develop the payoff matrix  $Y$ , since (2) is the only result pertaining to the substitution game that we shall need later.

With these preliminaries out of the way we survey the essential notation used in the authentication model.

Name	Set	Element	Variable
Source	$\mathcal{S}$	$s_i$	$S$
Message Space	$\mathcal{M}$	$m_j$	$M$
Encoding Rules	$\mathcal{E}$	$e_k$	$E$
Splitting Strategies		$\pi(m_j   s_i e_k)$	$\Pi$
Impersonation Strategy	$\mathcal{Q}$	$q_j$	$Q$

$P(X = x)$  probability that the random variable  $X$  takes the value  $x$ ,  
as for example  $P(M = m)$ ,  $P(S = s)$  or  $P(E = e)$ .

Name	Entropy
Source Distribution	$H(S)$
Message Distribution	$H(M)$
Coding Strategy	$H(E)$
Joint (message coding strategy source) Distribution	$H(MES)$

A	encoding matrix
X	impersonation payoff matrix
Y	substitution payoff matrix
XY	concatenated authentication payoff matrix
$v_I$	value of impersonation game on X (to opponent)
$v_S$	value of substitution game on Y (to opponent)
$P_d = v_G$	probability that opponent deceives the receiver: value of game on XY.

$|e_i| = \sum_{m \in \mathcal{M}} \chi(e_i, m)$       number of nonzero entries in the  $e_i$  row of  
either A or X.

$|m_j| = \sum_{e \in \mathcal{E}} \chi(e, m_j)$       number of nonzero entries in the  $m_j$  column of  
either A or X.

#### THE AUTHENTICATION CHANNEL BOUND

Our object in this paper is to derive channel bounds for the authentication channel. Several such bounds are easy.

##### Theorem 1.

$$(3) \quad P_d = v_G \geq \frac{\min_e |e_i|}{|\mathcal{M}|}.$$

##### Proof:

As has already been noted, the opponent has available as part of his strategy the choice of whether to impersonate the transmitter or to substitute messages, hence the value of the concatenated game is at least as large as the value of either game alone. We actually prove that for the impersonation game:

$$v_I \geq \frac{\min_e |e_i|}{|\mathcal{M}|}.$$

The payoff matrix for A is the  $|\mathcal{E}| \times |\mathcal{M}|$  (0,1) matrix X in which  $\chi(i, j) = 1$  if some state of  $\mathcal{S}$  is encoded into  $m_j$  by the encoding rule  $e_i$ , and 0 otherwise. If the transmitter/receiver are playing an optimal strategy E (probability that encoding rule  $e_i$  is played is  $P(E = e_i)$ ) and the opponent is impersonating the transmitter with an optimal strategy Q (probability that he sends  $m_j$  is  $q_j$ ) then the expected value to the opponent of impersonating with message  $m_j$  is

$$r_j = \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m_j)$$

and his expected payoff from playing strategy Q is simply the value of the game

$$v_I = \sum_{m \in \mathcal{M}} \{q(m) \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)\}.$$

Since  $v_I$  is the value of the game for the opponent, realized playing an optimal strategy Q, it is at least as large as the value realized by his playing any other strategy -- in particular, the uniform probability distribution of  $\mathcal{M}$ . Therefore,

$$v_I = \sum_{m \in \mathcal{M}} \{q(m) \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)\} \geq \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)$$

(4)

$$= \frac{1}{|\mathcal{M}|} \sum_{e \in \mathcal{E}} P(E = e) \sum_{m \in \mathcal{M}} \chi(e, m) = \frac{1}{|\mathcal{M}|} \sum_{e \in \mathcal{E}} |e| P(E = e).$$

The inequality is only weakened by replacing  $|e|$  by  $\min_{\mathcal{E}} |e|$ . Therefore,

$$v_G \geq v_I \geq \frac{\min_{\mathcal{E}} |e|}{|\mathcal{M}|}$$

as was to be shown. ■

#### Corollary:

Since  $\min_{\mathcal{E}} |e| \geq |\mathcal{S}|$

$$(5) \quad P_d = v_G \geq \frac{|\mathcal{S}|}{|\mathcal{M}|}.$$

#### Theorem 2.

Given an authentication system  $(\mathcal{S}, S, A)$  for which

$$(6) \quad v_G = \frac{\min_{\mathcal{E}} |e|}{|\mathcal{M}|};$$

in every optimal strategy, E, for the transmitter/receiver  $P(E = e) = 0$  for any encoding rule for which  $|e| > \min_{\mathcal{E}} |e|$ .

#### Proof:

As in the proof of Theorem 1 we use the fact that  $v_G \geq v_I$  and actually prove the conditions of the theorem for the impersonation game. From (4) we have

$$v_I = \sum_{m \in \mathcal{M}} q(m) \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) .$$

Assume that there is some encoding rule,  $e_j$ , for which  $|e_j| > \min_{\mathcal{E}} |e|$  and for which  $P(E = e_j) > 0$ . As noted before  $Q$  is an optimal strategy for the opponent and hence  $v_I$  is at least as great an expectation for him as he could achieve using any other strategy -- in particular the uniform probability distribution on  $\mathcal{M}$ .

$$v_I \geq v(\text{uniform}) = \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{M}|} \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)$$

$$= \sum_{e \in \mathcal{E}} P(E = e) \sum_{m \in \mathcal{M}} \frac{\chi(e, m)}{|\mathcal{M}|} = \sum_{e \in \mathcal{E}} P(E = e) \frac{|e|}{|\mathcal{M}|} > \frac{\min_{\mathcal{E}} |e|}{|\mathcal{M}|}$$

if  $P(E = e) > 0$  for any  $e \in \mathcal{E}$  for which  $|e| > \min_{\mathcal{E}} |e|$ . ■

Corollary:

If for an authentication system  $(\mathcal{S}, S, A)$

$$v_G = \frac{|\mathcal{S}|}{|\mathcal{M}|}$$

which by Theorem 1 can only happen if  $\min_{\mathcal{E}} |e| = |\mathcal{S}|$ , then every optimal strategy for the transmitter/receiver,  $E$ , has  $P(E = e) = 0$  for any encoding rule for which  $|e| > |\mathcal{S}|$ .

Another way of stating the conclusion of the Corollary is that if  $v_G = |\mathcal{S}|/|\mathcal{M}|$  no splitting occurs in any encoding rule occurring in an optimal strategy! It is worth remarking that

$$v_G = \frac{\min_{\mathcal{E}} |e|}{|\mathcal{M}|}$$

does not imply that splitting does not occur in any of the encoding rules that occur in  $\mathcal{E}$ . What is true, by Theorem 2, is that in this case all of the encoding rules that occur (with positive probability) in an optimal strategy use the same number of messages.

Several other channel capacity theorems of similar flavor can be proven, however we now turn to our primary object in this paper; establishing bounds on the authentication channel in terms of the various entropies on the primary variables. A trivial bound can be given in terms of  $H(E)$ . Since  $H(E)$  is the total equivocation that the opponent has as to which encoding rule is being used by the transmitter/receiver, and since he could deceive the receiver with certainty if he only knew the rule they had chosen, we have

$$(7) \quad \log P_d = \log v_G \geq -H(E)$$

(7) isn't a particularly useful result since as we shall see later there is a much stronger bound in terms of  $H(E)$ . The bound of the following theorem is the main result on which the theory of authentication is based.

Theorem 3. (Authentication Channel Capacity)

$$(8) \quad \log P_d \geq H(MES) - H(E) - H(M)$$

Proof:

Let  $P(M = m)$  be the probability that message  $m$  will be observed in the channel when states of the source occur according to the probability distribution  $S$  and are encoded by the transmitter with an encoding rule chosen from  $\mathcal{E}$  with probability distribution  $E$ , employing splitting strategies  $\Pi$ .  $P(M = m)$  is formally

$$(9) \quad P(M = m) = \sum_{(e,s) \in \mathcal{E} \times \mathcal{S}} P(M = m, E = e, S = s)$$

or equivalently by

$$(10) \quad P(M = m) = \sum_{(e,s) \in \mathcal{E} \times \mathcal{S}} P(M = m, E = e, S = s) \chi(e, m)$$

$$\text{where } \chi(e, m) = \begin{cases} 1 & \text{if some state of the source can be} \\ & \text{encoded into } m \text{ using encoding rule } e \\ 0 & \text{otherwise} \end{cases}$$

The formal sum (10) has the same value as (9) since

$$P(M = m, E = e, S = s) \neq 0 \rightarrow \chi(e, m) = 1$$

The converse need not be true, i.e.,  $\chi(e, m) = 1$  can hold while  $P(M = m, E = e, S = s) = 0$ , either because some  $s'$ , other than the  $s$  in  $P(M = m, E = e, S = s)$  is encoded into  $m$  by  $e$ , or else that the state occurring in  $P(M = m, E = e, S = s)$  could be encoded into  $m$  and some other message(s) under  $m$ , but that the splitting rule used by the transmitter never uses  $m$ .  $\chi(e, m)$  is the authentication function on  $\mathcal{M}$  since the receiver will accept a message  $m$  when encoding rule  $e$  has been selected if and only if  $\chi(e, m) = 1$ .

The joint probability  $P(M = m, E = e, S = s)$  can be represented as the product of the conditional probability that  $m$  will be sent given that state  $s$  occurred and that encoding rule  $e$  is being used  $\Pi(m|e, s)$ , times the independent probabilities that these events occur.

$$(11) \quad P(M = m) = \sum_{(e,s) \in \mathcal{E} \times \mathcal{S}} P(E = e) \chi(e, m) P(S = s) \pi(m|e, s) .$$

We now wish to restrict the domain from the Cartesian product  $\mathcal{E} \times \mathcal{S}$  to only  $\mathcal{E}$  by using the inverse mapping to  $e$ ;  $D(e, m)$ ,  $\chi(e, m)$  was introduced in (2) to make this possible,

$$(12) \quad P(M = m) = \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) P(S = D(e, m)) \pi(m|e, D(e, m))$$

since

$$\chi(e, m) \pi(m|es) = 0 \quad \text{unless} \quad D(e, m) = s .$$

Define a probability distribution  $W(m) = \{w_e(m)\}$  on  $e \in \mathcal{E}$  for every  $m \in \mathcal{M}$ :

$$(13) \quad w_e(m) = \frac{P(E = e) \chi(e, m)}{\sum_{e^* \in \mathcal{E}} P(E = e^*) \chi(e^*, m)}$$

$w_e(m)$  is well defined since every  $m \in \mathcal{M}$  is acceptable to the receiver for at least one choice of an encoding rule. Also  $\sum_{e \in \mathcal{E}} w_e(m) \chi(e, m) = 1$ . Multiplying the summand in (12) by

$$\frac{\sum_{e^* \in \mathcal{E}} P(E = e^*) \chi(e^*, m)}{\sum_{e^* \in \mathcal{E}} P(E = e^*) \chi(e^*, m)}$$

we obtain

$$(14) \quad P(M = m) = \sum_{e \in \mathcal{E}} w_e(m) \left\{ \left[ \sum_{e^* \in \mathcal{E}} P(E = e^*) \chi(e^*, m) \right] P(S = D(e, m)) \pi(m|e, D(e, m)) \right\} .$$

We now wish to form  $-P(M = m) \log P(M = m)$  on both sides of (14) as a first step to calculating the entropy  $H(M)$  of the messages observed in the channel. Formally,

$$(15) \quad -P(M = m) \log P(M = m) = - \left[ \sum_{e \in \mathcal{E}} w_e(m) \{ \dots \} \right] \log \left[ \sum_{e \in \mathcal{E}} w_e(m) \{ \dots \} \right] .$$

Noting that  $-x \log x$  is concave downwards, we use Jensen's inequality -- which says that if  $g(x)$  is a concave function on  $(a, b)$ , and if  $\{x_i\}$  are arbitrary real arguments,  $a < x_i < b$ , then for any set of positive weights  $w_i$  where  $\sum w_i = 1$ ;

$$g(\sum w_i x_i) \geq \sum w_i g(x_i)$$

to replace the equality in (15) with an inequality.

Let  $x = \{\dots\}$  in (7):

$$(16) \quad -P(M = m) \log P(M = m) \geq - \sum_{e \in \mathcal{E}} w_e(m) \{\dots\} \log \{\dots\} .$$

By canceling the sum  $\sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)$  between the denominator of  $w_e(m)$  and  $\{\dots\}$ , and by splitting the logarithm of the product in  $\{\dots\}$  into the sum of three logarithms, we get

$$(17) \quad \begin{aligned} -P(M = m) \log P(M = m) \geq & - \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) P(S = D(e, m) \pi(m) e, D(e, m)) \\ & \times \left[ \log \left( \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) \right) + \log P(S = D(e, m)) + \log \pi(m|e, D(e, m)) \right] \end{aligned}$$

Now, we make use of the game model for the authentication channel to bound (17) below. The value of the impersonation game,  $v_I$ , is

$$(18) \quad v_I = \max_{m \in \mathcal{M}} \sum_{e \in \mathcal{E}} P(E^* = e) \chi(e, m) \geq \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m)$$

where  $E^*$  is an optimal strategy for the transmitter/receiver and  $E$  is an arbitrary strategy. Inequality (18) is at worst weakened through replacing

$$\sum P(E = e) \chi(e, m)$$

in [...] with the maximum value it can have for any choice of  $m$ . Summing both sides of (18) over all  $m \in \mathcal{M}$ , we get

$$H(M) = - \sum P(M = m) \log P(M = m)$$

on the left and the expression in (19) on the right:

$$(19) \quad \begin{aligned} H(M) \geq & - \sum_{m \in \mathcal{M}} \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) P(S = D(e, m)) \pi(m|e, D(e, m)) \\ & \times [\log v_I + \log P(S = D(e, m)) + \log \pi(m|e, D(e, m))] . \end{aligned}$$

Since  $\log v_I$  is a constant it can be moved through the double summation to give

$$\log v_I \sum_{m \in \mathcal{M}} \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) P(S = D(e, m)) \pi(m|e, D(e, m))$$

Using (12), the summand can be replaced by  $P(M = m)$

$$\log v_I \sum_{m \in \mathbb{M}} P(M = m) = \log v_I$$

so that (19) becomes

$$(20) \quad H(M) \geq -\log v_I - \sum_{m \in \mathbb{M}} \sum_{e \in \mathcal{E}} P(E = e) \chi(e, m) P(S = D(e, m)) \\ \times \pi(m|e, D(e, m)) [\log P(S = D(e, m)) + \log \pi(m|e, D(e, m))] .$$

It has already been noted that

$$\chi(e, m) \pi(m|e, s) = 0$$

unless  $D(e, m) = s$ , therefore (20) can be rewritten in the form

$$(21) \quad H(M) \geq -\log v_I - \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} \sum_{\substack{m \in \mathbb{M} \\ D(e, m) = s}} P(E = e) P(S = s) \pi(m|e, s) \\ \times \{\log P(S = s) + \log \pi(m|e, s)\}$$

or

$$(22) \quad = -\log v_I - \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} P(E = e) P(S = s) \log P(S = s) \\ - \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} P(E = e) P(S = s) \sum_{\substack{m \in \mathbb{M} \\ D(e, m) = s}} \pi(m|e, s) \log \pi(m|e, s)$$

since

$$\sum_{\substack{m \in \mathbb{M} \\ D(e, m) = s}} \pi(m|e, s) = 1 .$$

Moving the summation over  $\mathcal{S}$  through  $P(E = e)$ , we obtain

$$(23) \quad H(M) \geq -\log v_I + \sum_{e \in \mathcal{E}} P(E = e) H(S) + \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} P(E = e) P(S = s) H(M|ES)$$

$$(24) \quad = -\log v_I + H(S) + H(M|ES) .$$

Using the entropy identity



$$H(A|B) = H(AB) - H(B)$$

(16) becomes

$$(25) \quad \log v_I \geq H(S) + H(MES) - H(ES) - H(M) .$$

But

$$H(ES) = H(E|S) + H(S) = H(E) + H(S)$$

since E and S are independent. Therefore

$$\log v_I \geq H(MES) - H(E) - H(M) .$$

The conclusion of the theorem follows from the earlier result that

$P_d = v_G \geq \max(v_I, v_S)$ , so that

$$(26) \quad \log P_d = \log v_G \geq \log v_I \geq H(MES) - H(E) - H(M)$$

as was to be shown. ■

The hard work is now completed. A variety of useful equivalent expressions can be derived from (26) using simple identities from information theory, for the cases of authentication either with or without secrecy. We illustrate the technique in Theorem (4) for the case of authentication with secrecy: i.e., the opponent does not know the state of the source observed by the transmitter. This, of course, only matters when the opponent elects to substitute messages rather than to impersonate the transmitter.

#### Theorem 4.

$H(MES) - H(E) - H(M)$  is equivalent to any of the following eight entropy expressions.

	X	Equivalent Form
(27)	ES	$H(M ES) + H(S) - H(M)$
(28)	MS	$H(E MS) - H(E) + H(MS) - H(M)$
(29)		or $H(E MS) - H(E) + H(S M)$
(30)	ME	$H(E M) - H(E)$
(31)		or $H(M E) - H(M)$
(32)	S	$H(ME S) + H(S) - H(E) - H(M)$
(33)	E	$H(MS E) - H(M)$
(34)	M	$H(ES M) - H(E)$

Proof:

The proof in each case proceeds by splitting the argument in the entropy  $H(MES)$  through conditioning the joint probability on  $X$  and then using simple identities to reduce the resulting expressions. The derivation of (27) is typical.

$$\begin{aligned} H(MES) &= H(M|ES) + H(ES) \\ &= H(M|ES) + H(E|S) + H(S) \\ &= H(M|ES) + H(E) + H(S) \end{aligned}$$

since  $E$  and  $S$  are independent random variables. Hence

$$H(MES) - H(E) = H(M) = H(M|ES) + H(S) = H(M)$$

as was to be shown, etc. ■

Using the results of Theorem 4 it is possible to derive some (generally) weaker but enlightening channel bounds. We first note that the total effective equivocation to the opponent playing the substitution game but without knowledge of the source state, i.e., authentication with secrecy is no greater than  $H(E|M)$  and as remarked earlier, the opponent's total effective equivocation if he knows the source state, i.e., authentication without secrecy, is at most  $H(E|MS)$ .

Theorem 5.

For authentication with secrecy

$$(35) \quad \log v_G \geq -\frac{1}{2} H(E)$$

while for authentication without secrecy

$$(36) \quad \log v_G \geq -\frac{1}{2} \{H(E) - H(MS) + H(M)\} = -\frac{1}{2} \{H(E) - H(S|M)\}$$

Proof:

For authentication with secrecy

$$(37) \quad \log v_G \geq \min\{\log v_I, -H(E|M)\}$$

while for authentication without secrecy

$$(38) \quad \log v_G \geq \min\{\log v_I, -H(E|MS)\} \quad .$$

In either (37) or (38) the bounds derived in Theorems 3 and 4 on the value of the impersonation game can be substituted, since the opponent's impersonation strategy is independent of whether he plays substitution with or without secrecy. Replacing the minimum on the right-hand side of the inequality by the average of the two bracketed terms either weakens the inequality if the terms are not identical or leaves it unaffected if they are. Therefore for authentication with secrecy, replacing  $v_I$  with the bound (30) in (37) we get

$$\log v_G \geq \frac{1}{2} \{H(E|M) - H(E) - H(E|M)\} = -\frac{1}{2} H(E)$$

and similarly by replacing  $v_I$  with the bounds (28) or (29) in (38) we get

$$\begin{aligned} \log v_G &\geq \frac{1}{2} \{H(E|MS) - H(E) + H(MS) - H(M) - H(E|MS)\} \\ &= -\frac{1}{2} \{H(E) - H(MS) + H(M)\} \end{aligned}$$

or

$$\begin{aligned} \log v_G &\geq \frac{1}{2} \{H(E|MS) - H(E) + H(S|M) - H(E|MS)\} \\ &= \frac{1}{2} \{H(E) - H(S|M)\} \end{aligned}$$

as was to be shown. ■

#### Corollary:

$$(39) \quad P_d = v_G \geq \frac{1}{\sqrt{|\mathcal{E}|}}$$

#### Proof:

$$H(E) \geq \log |\mathcal{E}|$$

with equality if and only if the transmitter/receiver's optimal strategy  $E$  is the uniform probability distribution on  $\mathcal{E}$ . The conclusion follows by substituting (39) into (35). ■

Bound (35) was first found by Gilbert, McWilliams and Sloan [7] under slightly more restrictive conditions and derived directly in the same generality used here by Simmons and Brickell in [6]. (35) is the bound based on  $H(E)$  promised earlier when the trivial bound in (7) was given.

## FOR EXAMPLE

In this section, in order to show the effects of secrecy on both the strategies of the participants and on the game values as well as to illustrate parameters such as splitting, etc., we discuss two small examples. Earlier we described an authentication system for which  $|E| = |M| = 4$ ,  $|S| = 2$  and for which the payoff matrix  $X$  was:

$$(40) \quad X = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix}$$

$X$  could also be the payoff matrix for many different authentication systems, one of which was exhibited before

$$(41) \quad A = \begin{vmatrix} s_1 & s_2 & 0 & 0 \\ s_1 & 0 & s_2 & 0 \\ 0 & s_2 & 0 & s_1 \\ 0 & 0 & s_2 & s_1 \end{vmatrix}.$$

One other such system is

$$(42) \quad A^* = \begin{vmatrix} s_1 & s_2 & 0 & 0 \\ s_2 & 0 & s_1 & 0 \\ 0 & s_1 & 0 & s_2 \\ 0 & 0 & s_2 & s_1 \end{vmatrix}.$$

In either case  $v_I = 1/2$  with an optimal strategy for either player being the uniform probability strategy on rows (transmitter) and on columns (opponent). If we consider only substitution with secrecy, then it makes no difference to the opponent whether the transmitter/receiver are using the authentication system  $(S, S, A)$  or  $(S, S, A^*)$ , since in either case when he sees a message he is faced with two possible encoding rules and hence with a choice between two equilikely messages to substitute -- one of which will be accepted and are rejected. His probability of success in either case is  $1/2$ , which is precisely what his chances of success in impersonating the transmitter would have been had he not waited to observe a message. Hence for authentication with secrecy  $P_d = 1/2$ . The situation is different however for authentication without secrecy. In this case for the system  $(S, S, A)$  the same arguments given for the authentication with secrecy case hold and  $P_d = v_G = v_I = v_S = 1/2$ . For the system  $(S, S, A^*)$  however, if the opponent waits to observe a message he will know with certainty which encoding rule the transmitter/receiver

have chosen and hence can substitute another message with certainty that not only will it be accepted as authentic by the receiver but that the receiver will be misinformed as a result. Therefore in this case

$$P_d = v_G = v_s = 1 > v_I = \frac{1}{2}.$$

Incidentally the system  $(\mathcal{S}, S, A)$  is perfect and is also an instance in which equality holds in (39):

$$v_G = \frac{1}{2} = \frac{1}{\sqrt{|\mathcal{E}|}}.$$

We conclude by showing another example in which equality holds in (39) and in which, in addition, splitting is essential (for the transmitter/receiver) to hold the opponent to the game value  $P_d = 1/\sqrt{|\mathcal{E}|}$ . In order to have a concise description of  $(\mathcal{S}, S, A)$  we introduce a notation for  $A$ .  $\mathbb{M}$  is partitioned into disjoint parts -- three in the example -- and the elements in each part indexed. The encoding rules will be of a special type (Cartesian) that encode a state of the source only into the messages in a particular part. In the example  $|\mathcal{S}| = 3$ ,  $|\mathbb{M}| = 12$  and  $|\mathcal{E}| = 16$ . The partition of  $\mathbb{M}$  is into 4, 4 and 8 elements, indexed 1, 2, 3, 4; 1, 2, 3, 4 and 1, 2, 3, 4, 5, 6, 7, 8, respectively. The states of the source are assumed to be equiprobable.

	$s_1$	$s_2$	$s_3$
$A =$	1	1	1,2
	1	2	3,4
	1	3	5,6
	1	4	7,8
	2	1	3,8
	2	2	1,7
	2	3	2,4
	2	4	5,6
	3	1	5,7
	3	2	2,6
	3	3	1,8
	5	4	3,4
	4	1	4,6
	4	2	5,8
	4	3	3,7
	4	4	1,2

Encoding rule  $e_1$  says that source state  $s_1$  will be encoded into message 1 of part 1, state  $s_2$  into message 1 of part 2 and state  $s_3$  into either message 1 or message 2 of part 3, etc. The unique optimal strategy,  $E$ , for the transmitter/receiver is the uniform probability distribution  $p(E = e_1) = 1/16$  with uniform splitting; i.e., if  $e_1$  is being used and state  $s_3$  occurs, then a fair coin would be tossed to decide

whether message 1 or 2 of part 3 was to be sent, etc. Against strategies S and II, the value of the game is

$$P_d = v_G = \frac{1}{\sqrt{|E|}} = \frac{|S|}{|M|} = \frac{3}{12} = \frac{1}{4}.$$

and the game is perfect. Although it isn't quite obvious, it is easy to show that it doesn't matter to the opponent whether he chooses to impersonate the transmitter or to wait and observe a message and then substitute another message; in either case if he plays optimally his chance of success will be  $1/4$ . Note that in this example while the opponent is faced with two bits of equivocation irrespective of whether he impersonates or substitutes, i.e.,  $v_I = v_S = 1/4$ , that the equivocation about the source state is only  $\log_2 3 = 1.585$  bits, or  $P(S = s) = 1/3$  for any  $s \in S$ . Thus while the opponent could guess the state of the source with a probability of success of  $1/3$  he could only guess at a message to communicate a state with probability  $1/4$ . If one considers what the channel bound theorem says, this is no paradox and  $P_d$  can be made as small as desired, even for a one-bit source in which  $P(S = s) = 1/2$ . This example, incidentally, is one of the smallest illustrating an infinite class of perfect authentication systems [5] with essential splitting.

#### CONCLUSION

In this paper we have proven that the bounds on the authentication channel are precisely what one would intuitively expect (and hope for), namely that the difference between the amount of information transmitted through the channel and that needed by the receiver to resolve his equivocation about the source state can be used to authenticate the message, and conversely that no better result can be achieved. We also exhibited small examples demonstrating that it is possible to use all of this residual information to confound the opponent, i.e., that the channel bounds are sharp.

#### REFERENCES

1. A. D. Wyner, "The Wire-tap Channel," The Bell System Technical Journal, Vol. 54, No. 8 (Oct. 1975), pp. 1355-1387.
2. G. J. Simmons, "A Preliminary Report on a Theory of Authentication," Proceedings of the IEEE National Electronics Conf., Chicago, IL (Oct. 28-29, 1981), pp. 315-318.
3. G. J. Simmons, "Verification of Treaty Compliance -- Revisited," Proceedings of the IEEE 1983 Symposium on Security and Privacy, Oakland, CA (Apr. 25-27, 1983), pp. 61-66.
4. G. J. Simmons, "A System for Verifying User Identity and Authorization at the Point-of-Sale or Access," Cryptologia, Vol. 8, No. 1, January 1984, pp. 1-21.

5. G. J. Simmons, "Message Authentication: A Game on Hypergraphs," Proceedings of the 15th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, March 5-8, 1984, (to appear).
6. E. F. Brickell, "A Few Results in Message Authentication," Proceedings of the 15th Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, LA, March 5-8, 1984, (to appear).
7. E. N. Gilbert, Mrs. F. J. MacWilliams, and N. J. A. Sloane, "Codes which Detect Deception," The Bell System Technical Journal, Vol. 53, No. 3 (March 1974), pp. 405-414.

# NEW SECRET CODES CAN PREVENT A COMPUTERIZED BIG BROTHER

David Chaum

Center for Mathematics and Computer Science (CWI)  
Kruislaan 413  
1098 SJ Amsterdam, The Netherlands

## ABSTRACT

As the use of computers becomes more pervasive, they are capturing increasingly more revealing data about our habits, lifestyles, values, whereabouts, associations, political and religious orientation, etc. The current approach, which requires individuals to identify themselves in relationships with organizations, allows records of all an individual's relationships to be linked and collected together into a dossier or personal profile. Even though such profiles are too extensive to evaluate manually on a mass basis, automated evaluation is becoming increasingly feasible.

A new approach prevents linking of such data, by allowing individuals to conduct relationships under different account numbers or "digital pseudonyms." The pseudonyms are created by a physical random process within a credit-card-sized computer carried by the individual. The card has no secrets from the individual or structure unmodifiable by the individual; it is merely a computer that acts on the individual's behalf and provides a convenient interface.

New cryptographic protocols provide security for both individuals and organizations against abuses by the other. A comprehensive set of three types of consumer transactions can be conducted, each using a different protocol. A communication protocol allows individuals to send and receive confidential and authenticated messages under pseudonyms. But even the tapping of all communication channels and the cooperation of all organizations does not allow messages to be traced to an individual. A payments system protocol allows an individual to pay or be paid, using an account maintained under a pseudonym with a bank. But even cooperation between the bank and other parties to payments does not allow payments to be traced to the individual's account. A credentials protocol allows digitally signed credential statements issued to an



individual under one pseudonym to be transformed into statements that can be shown on the individual's other pseudonyms. But the credentials shown to one organization do not allow tracing of the pseudonyms used with other organizations. All three protocols can be shown by simple mathematical proof to be unconditionally un-traceable, i.e. untraceable no matter how much computation is expended by tracing efforts.

This new approach may actually provide better protection against abuse by individuals, even in areas like consumer credit, social welfare, insurance, etc. than could be acceptably obtained under the current approach. Organizations may favor the new paradigm also because of reduced costs, reduced data maintenance and related exposure, and because of the opportunities for improved good will with advanced computerization. For individuals, the new paradigm offers greater convenience because they can select the card computer that suits them best, the full capabilities of a lost card can conveniently be restored into a replacement card, and the card can protect itself against use by anyone other than its owner. As the public becomes more aware of and familiar with the extent and possibilities of emerging information technology, appreciation of these advantages may grow. Of course an essential advantage of the new approach to individuals is the potential it offers them for regaining monitorability and control over the maintenance and use of information about themselves by others.

## REFERENCES

The presentation surveyed the three references. An invitation for journal publication of a survey article has been accepted.

- (1) Chaum, D., "The Dining Cryptographers Problem: Unconditional Sender and Recipient Anonymity," submitted for publication.
- (2) Chaum, D., "Privacy Protected Payments: Unconditional Payer and/or Payee Anonymity," submitted for publication.
- (3) Chaum, D., "Showing Credentials Without Identification: Transferring Signatures Between Unconditionally Unlinkable Pseudonyms," submitted for publication.

## Fair Exchange of Secrets

(extended abstract)

*Tom Tedrick \**

Computer Science Division  
573 Evans Hall  
University of California  
Berkeley, California 94720

Electronic mailing address:  
tedrick@berkeley

**Key Words and Phrases:** Exchange of Secret Keys, Contract Signing, Exchange of Secrets, Fractions of a Bit, Oblivious Transfer, Cryptographic Protocols

### Abstract

We consider two problems which arose in the context of "The Exchange of Secret Keys" (see [1]).

(1). In the original protocol, one party may halt the exchange and have a 2 to 1 expected time advantage in computing the other party's secret. To solve this problem, when there is a particular point in the exchange where this time advantage may be critical, we presented at CRYPTO 83 (see [5]), a method for exchanging "fractions" of a single bit.

In this paper we extend the method so as to apply it to all bits to be exchanged, and show how it can be used in a more abstract setting (as in [2]).

(2). We also present a solution to the problem of how to ensure a fair exchange of secrets when one party in the exchange is "risk seeking", while the other is "risk-adverse".

### Notation:

We use  $2^k$  to signify exponentiation, i.e.  $2^k$  represents 2 raised to the  $k$ th power.

### Introduction:

The following scenario occurs in both [1] and [2].

There are 2 parties A (Alice) and B (Bob).

Alice holds  $n$  pairs of  $m$  bit long secrets

$\langle a(1,1), a(1,2) \rangle, \langle a(2,1), a(2,2) \rangle, \dots, \langle a(n,1), a(n,2) \rangle$

Bob knows exactly one secret from each pair, while Alice does not know which one he knows (this condition can be achieved using an "oblivious transfer" protocol as in [2] and [4]).

Similarly Bob has  $n$  pairs of secrets denoted

$\langle b(1,1), b(1,2) \rangle, \dots, \langle b(n,1), b(n,2) \rangle$

Alice knows exactly one secret from each pair, etc.

Each party is eager to know both elements of any pair of his counterpart's secrets. In Blum's "Exchange of Secret Keys" such knowledge allows one to factor (i.e. obtain the secret key of an RSA/Rabin or Goldwasser/Micali public key crypto-system). In the Even/Goldreich/Lempel "Randomized Protocol for Signing Contracts" knowledge of a pair constitutes having a signature to the contract.

We assume that computing a secret can only be done by an exhaustive search of the secret space (the set of  $m$ -bit long strings).

---

\*Research sponsored in part by the National Science Foundation Grant MCS 82-04506

Both Blum and Even/Goldreich/Lempel apply the following protocol, in order to reach concurrent knowledge of one pair of secrets:

For  $k=1$  to  $m$  do

- (1). Alice sends the  $k$ th bit of each  $a(i,j)$
- (2). Bob sends the  $k$ th bit of each  $b(i,j)$

Note that in order to prevent the counterpart from getting any pair of secrets, a dishonest party must send incorrect bits for at least one element in each pair. But the chance of getting away with this is  $1/2^n$ .

If both parties follow the protocol properly each can compute a complete pair of their counterpart's secrets in about the same amount of time. However some problems arise which we discuss in the following sections.

### First Problem:

If Bob halts the protocol after Alice sends the  $k$ th bits of her secrets then Bob has a 2 to 1 computational advantage. He needs only search through a subset of  $2^{m-k}$  possible secrets to compute a pair, while Alice needs search a subset of size  $2^{m+1-k}$  (twice as large).

In [5] we discussed several methods of exchanging fractions of a bit when there is a key bit that is crucial. Micali/Luby/Rackoff have also written a very nice paper on exchanging a secret bit using a different approach (see[3]).

We here extend one of the methods in [5] so as to carry it out throughout the exchange, keeping Bob's computational advantage at any point below a predetermined amount.

### The method (an example)

We first illustrate the method by giving an example.

Bob and Alice agree that the maximum computational advantage will be 5 to 4 (instead of 2 to 1 as in the original protocol).

For each  $a(i,j)$ , Alice stores the strings

```
000
001
010 *
011
100
101
110
111
```

Exactly one of these strings corresponds correctly to the first three bits of  $a(i,j)$ , say 010 (marked with a \* for reference).

Bob acts similarly.

Now a series of exchanges takes place.

For each  $a(i,j)$  Alice sends the message:

the next three bits of  $a(i,j)$  are not xyz (say not 101 for example).

Bob responds similarly.

Note that after Alice sends her messages Bob has first an 8 to 7, then a 7 to 6, then a 6 to 5, then a 5 to 4 edge (in the ratio between the size of the secret space Alice has to search and the space Bob has to search).

When only half the original strings remain, say for example

001  
010 \*  
110  
111

8 new strings (for each  $a(i,j)$  etc.) are created by adding 0 or 1 to the old strings. We get:

0010  
0011  
0100  
0101 \*  
1100  
1101  
1110  
1111

Note again that exactly one of these strings corresponds to the correct first 4 bits of  $a(i,j)$ .

The exchange then takes place again until 4 strings remain (for each  $a(i,j)$ ), 8 new strings are created, etc.

Note that the maximum computational advantage for Bob is 5 to 4.

Note that the chance of getting caught cheating by sending incorrect strings is exactly the same as in the original protocol: each time incorrect information is sent the chance of being detected is 50%.

#### **A more formal description of the method:**

- (1). Decide on an acceptable integer  $k$ , where the maximum computational advantage will be  $(2^k)+1$  to  $2^k$ .
- (2). For each  $a(i,j)$  and  $b(i,j)$  store the  $2^{(k+1)}$  strings of length  $k+1$ .
- (3). Repeat until done:

For  $x=1$  to  $2^k$  do:

Alice sends a string, for each  $a(i,j)$

Bob sends a string, for each  $b(i,j)$

End {For}

Alice and Bob create  $2^{(k+1)}$  strings, for each  $a(i,j)$  and  $b(i,j)$ , by adding 0 or 1 to the  $2^k$  unspent strings.

End {Repeat}

#### **Time/space complexity of the method:**

With  $k$  chosen as above, there are  $o(n*(2^k))$  strings of length  $\leq m$  to be stored. So memory needed is  $o(n*m*(2^k))$ .

Time needed is  $o(n*(m^2)*(2^k))$ . We present later a slightly more complicated version of the method which may require an additional  $\log(m)$  factor.

#### **Suggested modification of "The Exchange of Secret Keys"**

Shamir/Goldreich have announced a method for breaking the original exchange of secret keys protocol. We suggest that the protocol should be modified in two ways:

- (1). In the original protocol  $a(i,1)$  and  $a(i,2)$  are distinct square roots of some quadratic residue  $X_i$  modulo Alice's public key, where Bob chooses  $X_i$ . We suggest that Alice should choose  $X_i$  at random and send Bob a root via oblivious transfer (see [4]).

(2). In the original protocol Alice sends the bits of  $a(i,j)$  in order, first, second, third ... . Instead the location of the next bit to be sent should be chosen randomly from the unused locations. The method described above for sending less than a bit fits well into this type of scheme and will be described briefly by means of an example.

**The method modified to be more random looking:**

Suppose we set a 5 to 4 advantage limit.

For each  $a(i,j)$  we create 8 strings (say the bit length  $m=20$ ). First choose a random location, say 5. Create 2 strings

```
---- 0 -----
---- 1 -----
```

Note that exactly one of these strings corresponds correctly to the bits of  $a(i,j)$ .

For each string create 2 new strings by choosing a random location and filling it with 0 or 1.

```
---- 0 ----- 0 ----
---- 0 ----- 1 ----
---- 1 ----- 0 ----
---- 1 ----- 1 ----
```

Note again that exactly one string corresponds to the real  $a(i,j)$ .

Repeat the above process for each of the 4 strings:

```
---- 0 -- 0 ----- 0 ----
---- 0 -- 1 ----- 0 ----
---- 0 ----- 1 -- 0 -
---- 0 ----- 1 -- 1 -
---- 1 ----- 0 ----- 0 ---
---- 1 ----- 0 ----- 1 ---
---- 1 ----- 1 - 0 -----
---- 1 ----- 1 - 1 -----
```

Again note that exactly one of these strings corresponds correctly to  $a(i,j)$ .

As in the first version, half the strings are exchanged. Then 8 new strings are created using the 4 remaining strings. For example if the remaining 4 strings were:

```
---- 0 -- 1 ----- 0 ----
---- 0 ----- 1 -- 1 -
---- 1 ----- 0 ----- 0 ---
---- 1 ----- 0 ----- 1 ---
```

we would create 8 new strings by choosing a random location in each old string and filling it with 0 or 1, getting say:

```
---- 0 -- 1 ----- 0 ---- 0
---- 0 -- 1 ----- 0 ---- 1
---- 0 ---- 0 ---- 1 -- 1 -
---- 0 ---- 1 ---- 1 -- 1 -
---- 1 - 0 -- 0 ----- 0 ---
---- 1 - 1 -- 0 ----- 0 ---
---- 1 ----- 0 0 ---- 1 ---
---- 1 ----- 0 1 ---- 1 ---
```

And so on. At each step exactly one string corresponds correctly to  $a(i,j)$ .

### **Risk seeking vs. Risk Adverse**

We showed that the expected time for computing a secret can be made reasonably equal for both parties. However this may not be enough to discourage "risk-seeking" parties which may try to exploit the fact that the variance is large.

Let  $T_a$  and  $T_b$  be random variables representing the time Alice and Bob need to compute each other's secret. We assume  $T_a$  and  $T_b$  have identical uniform distributions on some interval 1 to  $K$ . Neglecting insignificant terms (as we will throughout this analysis) we get  $E(T_a) = K/2$ .

Let  $Y = T_a - T_b$ . Then  $E(|Y|) = K/3$ . So there is a good chance that if Bob halts the protocol at a certain point, he will discover Alice's secret well before she discovers his.

Note  $E(Y^2) = (K^2)/6$ , or  $E(Y^2) = 1.5 * (E(T_a))^2$

One solution to this problem is to modify the nature of the secret. We take a large number, say  $X$ , old secrets. The new secret is defined to be knowledge of all  $X$  old secrets (there are interesting crypto-systems based on using a large number of keys, see [6], so this idea is not far fetched).

Note that  $T_a$  is now the sum of  $X$  uniformly distributed random variables. If  $Y = T_a - T_b$  as before, we find that  $E(Y^2) = (1.5 * (E(T_a))^2) / X$  as opposed to  $1.5 * (E(T_a))^2$  in the previous case. So the squared distance  $T_a - T_b$  is reduced by a factor of  $X$ . So the expected distance between  $T_a$  and  $T_b$  can be reduced to any level desired.

This is somewhat analagous to flipping a silver dollar as opposed to flipping 100 pennies ... If Alice gets the heads and Bob gets the tails then in each case they expect to get 50 cents, but in the first case the variance is larger.

Overall costs are multiplied by a factor of  $X$  if this method is used.

### **Acknowledgements:**

Oded Goldreich made many helpful suggestions.

### **References:**

- [1]. Blum, M. "How to Exchange Secret Keys", ACM Transactions on Computer Systems, 1983.
- [2]. Even, Goldreich, Lempel "A Randomized Protocol for Signing Contracts"
- [3]. Micali, Rackoff, Luby "The MiRackoLus Exchange of a Secret Bit", 1983 FOCS
- [4]. Peralta, Berger, Tedrick "A Provably Secure Oblivious Transfer", Eurocrypt 84
- [5]. Tedrick, T. "How to Exchange Half a Bit", CRYPTO 83
- [6]. Tedrick, T. "Some Advantages of Using Many Keys in Public Key Encryption Protocols"

CRYPTOPROTOCOLS: SUBSCRIPTION TO A PUBLIC KEY, THE SECRET  
BLOCKING AND THE MULTI-PLAYER MENTAL POKER GAME  
(extended abstract)

Mordechai Yung

Department of Computer Science

Columbia University

New York, N.Y. 10027

ABSTRACT

Investigating the capabilities of public key and related cryptographic techniques has recently become an important area of cryptographic research. In this paper we present some new algorithms and cryptographic protocols (*Cryptoprotocols*) which enlarge the range of applications of public key systems and enable us to perform certain transactions in communication networks. The basic cryptographic tools used are Rabin's *Oblivious Transfer Protocol* and an algorithm we developed for *Number Embedding* which is provably hard to invert.

We introduce the protocol *Subscription to a Public Key*, which gives a way to transfer keys over insecure communication channels and has useful applications to cryptosystems. We develop the *Secret Blocking Protocol*, specified as follows : 'A transfers a secret to B, B can block the message. If B does not block it, there is a probability  $P$  that he might get it. ( $1/2 \leq P < 1$ , where we can control the size of  $P$ ). A does not know if the message was blocked (but he can find out later)'.

The classic cryptotransaction is the *Mental Poker Game*. A cryptographically secure solution to the *Multi Player Mental Poker Game* is given. The approach used in constructing the solution provides a general methodology of provable and modular *Protocol Composition*.

## 1. INTRODUCTION

*Complexity-based cryptography* has two major areas of application: *Public Key Cryptosystems* [7] [15], to provide secure and authenticated communication, and Cryptographic Transactions, *Cryptotransactions* for short, to enable simulation of certain activities in communication [4] [6] [17]. These activities, while easily done face to face seem impossible to perform through the use of a communication network.

In this paper we present some new Cryptoprotocols to be used both for increasing security and flexibility of Public-Key Cryptosystems and as tools for implementing Cryptotransactions. The security of these protocols is based on the intractability of the factorization problem. The basic cryptographic tools used are Rabin's *Oblivious Transfer Protocol* and an algorithm for *number embedding* which is provably hard to invert. The results reported here were motivated by Blum's paper [4] and are based on [19].

We introduce the protocol *Subscription to a Public Key*, used for transferring keys over insecure communication channels and which has useful applications for cryptosystems. We then develop the *Secret Blocking Protocol*, specified as follows : "A transfers a secret to B. B can block the message. If B does not block the message he gets it with probability  $= P$ , where  $1/2 \leq P < 1$ , and we can control the size of  $P$ . A does not know if the message was blocked, but he can find out later".

The classic cryptotransaction is the *Mental Poker Game*. The problem, proposed by Robert Floyd, is: 'Is it possible to play a fair poker game over the telephone ?' Shamir, Rivest and Adleman [17] proved that from an information theoretic point of view it is impossible to play the game. They showed, however that from a complexity theoretic point of view, the game can be played, using the one way commutative modular exponentiation function. Although their protocol is elegant and the number of players is unlimited, Lipton [10] showed that one can easily mark some subsets of cards using it. We present a cryptographically secure solution to the *Multi Player Mental Poker Game*. Different solutions to the Two Player Mental Poker Game have recently been obtained independently by Blum [5], and by Goldwasser and Micali [8]. Their solutions include a protocol for Two Player Card Dealing.

The approach used in constructing the solution gives a general methodology of provable and modular *Protocol Composition*.



## 2. Number Theoretic and Cryptographic Background

### 2.1. NUMBER THEORETIC ALGORITHMS AND PUBLIC KEY SYSTEMS

The *main assumption* is: *FACTORIZATION* of a number  $n=pq$ , where  $p$  and  $q$  are large (say 100-digit) prime numbers is *HARD* to solve. On the other hand, some number theoretic algorithms that we use are *EASY* (random polynomial time). These include the primality test, [18] [9], prime generation and root extraction of  $x^2(\text{mod } n)$  given the factors of  $n$  [12]. (For a survey of number theory and number theoretic algorithms see [11] [9] [1].) In the protocols presented in this paper, we need an underlying public key system in order to transmit encoded and signed messages and to hide information using one way functions. Either the RSA system [15], the Rabin system [12], or the Blum-Goldwasser system [3] can be used. (If we use RSA we add the assumption that RSA breaking is *HARD*)

### 2.2. RABIN'S OBLIVIOUS TRANSFER PROTOCOL

Rabin found a way to send a secret obviously, that is, the sender  $A$  does not know if the secret is successfully transmitted to  $B$ , the probability of success is  $1/2$ .

#### Protocol 1 -THE OBLIVIOUS TRANSFER:

- step 1* :  $A$  creates a number  $n = pq$ . (The prime factorization of  $n$  is the secret.)
  - step 2* :  $A \rightarrow B : "n"$ . ( $\rightarrow$  means 'sends to'.)
  - step 3* :  $B$  selects a random  $x$  and computes  $z = x^2(\text{mod } n)$ .  $B \rightarrow A : "z"$ .
  - step 4* :  $A$ , knowing the factorization of  $n$ , computes the 4 square roots of  $z = \{x, -x, y, -y\}$ . He selects at random one of them, calls it  $s$  and  $A \rightarrow B : "s"$
  - step 5* : If  $B$  receives  $y$  or  $-y$  he gets the secret, if he receives  $x$  or  $-x$  he does not.
- end {protocol 1}

**Theorem 1:** Given  $x, y \in \mathbb{Z}_n^*$  (that is  $x, y < n$  and do not divide  $n$ ),  $x \neq y \pmod{n}$ ,  $-x \neq y \pmod{n}$  and  $x^2 = y^2 \pmod{n}$ , there is an *EASY* algorithm for factoring  $n$ .

Based on the previous theorem we can prove the properties of the above protocol:

**Theorem 2:** Using the oblivious transfer protocol,  $B$  can factor  $n$  (get the secret) with a probability (virtually) equal to  $1/2$ .  $A$  can not know if he transferred the secret successfully.

### 2.3. ONE WAY NUMBER EMBEDDING

Number embedding is an algorithm which gets a number  $M$  as input and distributes it into some pieces of information  $EM(M)$  which hide  $M$ . Giving  $EM(M)$  does not compromise  $M$  because in order to recover the number from the available hiding information one has to solve a *HARD* problem.  $EM(M)$  can be recovered to one and only one number:  $M$ . In [4] Blum gave such an algorithm. Here we use polynomial interpolation to design a number embedding algorithm that is provably

HARD to recover.

**Algorithm 2 - EMBEDDING USING INTERPOLATION :**

*step 1:* Choose  $K$  (say  $K=10$ ) random 100-digit prime numbers  $p_i$ ,  $i=1,\dots,10$ .

*step 2:* Choose 10 random 99-digit prime numbers  $q_i$ ,  $i=1,\dots,10$ .

*step 3:* Construct a polynomial of degree 10:  $P \pmod{R}$ , where  $R$  is a large prime ( $R > p_i, q_i$ ,  $i=1,\dots,10$ ), by using the 11 interpolation points :  $(p_i, q_i)$   $i=1,\dots,10$ , and  $(0, M)$ .

*step 4:* Compute  $n_i = p_i q_i$  ( $n_i$  hides interpolation point  $i$ ). The embedding of  $M$  is the sequence consisting of:  $R$  (the modulus), the numbers  $n_i$ ,  $i=1,\dots,10$ , and a point  $(u, v)$  such that  $v = P(u)$ , where  $u$  is a random number different from 0 and the  $p_i$ 's.  
end {algorithm 2}

The Result of the Algorithm:

Given  $EM(M)$ , one has to factor the 10  $n_i$ 's to recover the unknown  $M$ . Factorization of any 9 of them does not help (see [16]). Given  $M_1 \neq M_2$ , we can embed both using the same  $n_i$ 's; only the additional random point  $(u, v)$  is different. Therefore we can prepare all the  $n_i$ 's before the communication.  $EM$  is a one way one to many random operator. Using the fact that generation of numbers of the form  $n = p q$  is easy, and the random polynomial algorithm [2] [13] for finding roots of polynomials over  $GF(R)$  we can show that recovering of  $M$  is polynomially equivalent to factorization. The reduction to factorization is given in the following theorem:

**Theorem 3:** If we can easily recover  $M$  from  $EM(M)$  (even in  $\epsilon$  of the times) we can easily factor numbers of the form  $n = p q$ .

Now consider the oblivious transfer protocol. If we want  $A$  to be able to check whether or not he gave  $B$  the secret then we use *Oblivious Transfer With Receipt*: When  $B$  sends  $z = x^2 \pmod{n}$  he also sends  $EM(x)$  which is the receipt which hides  $x$  unambiguously. The receipt also makes it possible to check that  $z$  was created by squaring an  $x \in Z_n^*$  and is not a 'special quadratic', a quadratic such that knowledge of any of its roots enables factorization. It was suggested in [14] overcoming this problem by sending  $K$  quadratics in step 3 from which  $B$  chooses  $K-1$  at random and asks  $A$  to send their roots first and then the protocol goes on with the remaining quadratic.

### 3. SUBSCRIPTION TO A PUBLIC-KEY

The problem is:  $A$  has a public-key  $E = (n, e)$ , based on  $n = p q$  (RSA [15], Rabin [12] or Blum-Goldwasser [3]).  $A$  wants  $B$  to subscribe to the key, namely to get the decryption key  $D = (n, d)$ . To solve this problem without compromising the key,  $A$  and  $B$  use the following cryptotransaction:

**Protocol 3 - SUBSCRIPTION TO A KEY**

*step 1* : A publicizes E.

*step 2* :

a. B chooses K random numbers (say  $K=10$ ):  $x_1, \dots, x_{10}$ .

b. B checks: if  $\gcd(x_i, n)$  is not trivial for some  $x_i$ , then STOP. (B got the key, the chance for this is virtually zero.)

Otherwise B computes  $z_i = x_i^2 \pmod{n}$ ,  $i=1, \dots, 10$ .  $B \rightarrow A : "z_i, i=1, \dots, 10"$ .

*step 3* :

a. A, knowing the factorization of n, computes the 4 square roots of  $z_i = \{ x_i, -x_i, y_i, -y_i \}$ ,  $i=1, \dots, 10$ .

b. A uses procedure SELECT to choose one of the roots, and calls it  $s_i$  (the SELECT process makes sure that if  $z_i$  is sent twice then the same  $s_i$  is chosen).

c.  $A \rightarrow B : "s_i, i=1, \dots, 10"$ .

end {protocol 3}

**Theorem 4:** The protocol "Subscription to a Public-Key" ensures:

1. B gets the decryption key with probability at least  $1-(1/2^{10})$ .
2. An eavesdropper cannot get information from the protocol which helps him factor n.

The above protocol has several applications to cryptosystems (e.g. distribution of a group key).

**4. ABSTRACTION OF THE "MENTAL POKER GAME"****4.1. SPECIFICATION OF THE GAME**

For A and B to play a fair "Mental Poker Game" we need the following protocols:

1. *A protocol for Dealing Cards.* The security and verifiability specifications contain some antagonistic requirements which make the problem interesting.
2. *Protocols for other game steps:* These include discarding cards from one's hand, opening a card, etc. in a secure and checkable way.
3. *A Protocol for the Game Management* which links all the game steps together into a complete game.

**4.2. DEFINITION OF CARD SETS**

We define sets which are changed dynamically during the game.

*ALL* - the set of all the cards which is the universal (ordered) set:  $\{1, 2, 3, \dots, 52\}$ .

*AHAND (BHAND)* - cards which are currently in the hand of A (B).

*AUSED (BUSED)* - cards which were thrown from the hand of A (B).

$ASAW = AHAND \cup AUSED$ ,  $BSAW = BHAND \cup BUSED$ ,

$SAW = ASAW \cup BSAW$ .

*AOPEN (BOPEN) {TOPEN}* - cards opened by A (by B) {to the table}.

$OPEN = AOPEN \cup BOPEN \cup TOPEN$ .

$DECK = ALL - \{SAW \cup OPEN\}$  - the cards currently in the deck.

$DECK_A = DECK \cup BSAW$  - cards that according to A's partial knowledge can still be in the deck.

$DECK_B = DECK \cup ASAW$  - possible deck according to B's partial knowledge.

#### 4.3. REPRESENTATION OF THE GAME

The game is fully represented by the card sets, so we can look at the game as a *Knowledge Set Transition System*. The Interpretation of the game as a formal system helps us to design it and to prove its properties, using formal inference about user knowledge and card sets.

*States* : States are positional vectors of sets which are subsets of  $ALL$ .

A game-state :  $GS = (DECK, AHAND, BHAND, ASAW, BSAW, OPEN)$ .

A special state is the illegal state which is a dead state.

The initial state is  $(ALL, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$

*Knowledge* : The player's partial knowledge of the game, at any moment of the game is also represented by a set vector. The set notation is augmented by the following: 1.  $?$  - an unknown set. 2.  $?_i$  - an unknown set of size  $i$ , where the size is the only knowledge about it. A's Partial-Knowledge (PK) of the game is:

$$- PK(A) = ( ?_{|DECK|}, AHAND, ?_{|BHAND|}, ASAW, ?_{|BSAW|}, OPEN )$$

*Transitions* : The transitions are the game steps {Dealing, Discarding, Opening, Opening from  $DECK$ }. Any illegal game step leads to the illegal state.

Our proof technique uses assertions on knowledge and card sets, showing for example that the following are game-invariant:  $ASAW \cap BSAW = \emptyset$ ,  $DECK \cap SAW = \emptyset$ ,

$ASAW \cap BSAW = \emptyset$ ,  $DECK_A \cap DECK_B = DECK$  and the fact that combining both players PK's gives the game state.

### 5. An Algorithm for Dealing Cards

#### 5.1. FIRST APPROXIMATION OF THE DEALING PROTOCOL

*The general idea*: When B draws cards from  $DECK$ , they are actually offered to him by A as follows:

A knows  $DECK_A = (DECK \cup BSAW) = ALL - \{ASAW \cup OPEN\}$ . Using this knowledge, A tries to transfer cards he has not yet seen without revealing which cards he is offering and without being able to know which cards are chosen by B. During the process of dealing B gets a card  $M \in DECK$ , this card is a random card from the deck. When B gets the  $j$  cards he needs  $M_1, \dots, M_j$  he is responsible for halting the dealing without trying to look at other cards in  $DECK$ . Then B updates :  $BHAND := BHAND \cup \{M_1, \dots, M_j\}$ ;  
 $BSAW := BSAW \cup BHAND$ .

# Protocol 4: THE DEALING OF CARDS PROTOCOL

- *step 1* :
  - a. A chooses  $M_1, \dots, M_{s_2}$  random 100-digit numbers to represent *ALL*
  - b. A computes  $f(ALL) = f(M_1), \dots, f(M_{s_2})$ ,  $f$ - is A's one way function.
  - c.  $A \rightarrow B$  : "  $f(ALL)$  "
- *step 2* : A tries to transfer cards from the  $l$  cards of  $DECK_A$ :
  - a. A embeds the cards in  $DECK_A$ .
  - b. Permutation choosing - (this sub-step is eliminated later, we need it just for the first approximation):
    - A chooses a random permutation of  $\{1, \dots, l\}$ :  $P_A$ , and hides it unambiguously in  $EM(P_A)$
    - $A \rightarrow B$  : "  $EM(P_A)$  "
    - B chooses a random permutation  $P_B$ ,  $B \rightarrow A$  : "  $P_B$  "
    - A computes  $P = (P_B \cdot P_A)$ . Let  $\{1, \dots, l\}$  be the order of  $DECK_A$  derived from the order of *ALL*.  $P$  is a random permutation of it, and B does not know what  $P$  is.
- *step 3* : A sends cards to B:
  - a.  $A \rightarrow B$  : "  $EM(M_{P(i)}), i = 1, l$  "
  - b. Oblivious transfers :  
 (The goal of this step is to let B factor the embeddings of cards. The permutation  $P$ , the probability of success of a single transfer and the merging of the transfers of the different cards, randomizes which cards are to be factored. This is Blum's idea for sending certified mail [4].)  
*begin loop* :  
   for  $j = 1$  to  $k$  {  $k$  is the size of each embedding }  
   for  $i = 1$  to  $l$  do :  
     A single OBLIVIOUS TRANSFER with RECEIPT  
     to enable factorization of  $n_{(P(i),j)}$   
   *end loop*.
  - c. *Getting a card* : During the above transfer process B factors all  $EM(M)$  so he gets  $M$ , then he computes  $f(M)$  and he knows which card  $M$  represents. If  $M \in BSAW$  nothing happened, else B adds  $M$  to his hand.
- *step 4* : After B gets the number of cards he needs, he halts the protocol by  
 $B \rightarrow A$  : " stop, I got  $j$  cards "  
*end {protocol 4}*

## The Remaining Problems:

1. It is possible that in step 4 player B must halt the process right after he took the last card he needed, and if he does not halt, he may see an extra card from *DECK* which he is not supposed to see. We will show a solution for this 'Halting

Problem'.

2. A knows a priori the order in which the cards are offered. Even if we let B choose which  $n_i$  he wants to try to factor in any order, there is still a bias. A knows at any moment (including the end of the dealing) the current probability with which any card is given, and different cards have different probabilities at least  $1-(1/l)$  of the time. The solution to problem 1 will solve problem 2 as well.

## 5.2. SECOND APPROXIMATION OF THE DEALING PROBLEM - THE SOLUTION TO THE 'PROTOCOL HALTING PROBLEM': "THE SECRET BLOCKING"

The purpose of this approximation is to explain the idea of 'The Secret Blocking'.

### *The Solution to the Problems:*

- 1. A (symmetrically B) has a set of public keys :

$$KEY_A = REALKEY_A \cup DUMMYKEY_A$$

$REALKEY_A$  is the set of keys for which A has both the encryption and the private decryption keys. For the dummy keys, A has only encryption keys and he can not decode messages encrypted by them. Half of the keys are real; half of them are dummy. (symmetrically  $KEY_B$  has the same subsets.) We assume *temporarily* that *KEYS* are given to the parties before the game by a *Judge* who knows which keys are real and which keys are dummies.

A (B) publishes all his encryption keys in a random order. B (A) can not know which keys are real, and which are dummies. We call these keys '*root-transfer keys*'.

- 2. the oblivious transfers in step 3 of protocol 4 are as follows:
  - a. Before A obviously transfers a root,  $B \rightarrow A$  : " use my root-transfer key  $K_i$  " and then  $A \rightarrow B$  : " a root  $s_i$  encrypted by this key ".
  - b. At first B chooses a key at random from  $KEY_B$ . If he chose a real key he may get the factorization with probability  $= 1/2$ , but if he chose a dummy key he gets information he can not decrypt. Hence the probability that B gets the factorization is  $1/4$ .
  - c. It is agreed that the halting of the protocol is at the end of the loop in step 3. After B gets all the cards he needs, he must continue the transfers until the end of the loop. For  $M_i$  which he has not yet recovered, he chooses a random number from  $EM(M_i)$  not yet factored, and for its root transfer he chooses a *random dummy-key*. Doing so he ensures that he gets information he can not decode and still he can not tell what  $EM(M_i)$  hides. Because of the random choice of root-transfer keys during the whole process A has no idea which information was blocked like this by B. This is "*The Secret Blocking*". The secret blocking also solves the second problem. What is actually done is : B

chooses a priori which embeddings of cards to try to take and which to block. Thus P (the random permutation of the offered cards) can be chosen by A alone.

- 3. We have a *Protocol for Open Replay of the Dealing* which is used for verification. When the game is over the *Judge* uses the receipts to replay the dealing and verifies that:
  - a. B got exactly the number of cards he needed, he got them from *DECK*, and he did not see any extra cards.
  - b. A always used the encryption key E which B asked him to use, and did not try to check a key's status by sending some other random message.

### 5.3. HOW TO ELIMINATE THE CENTRAL JUDGE: THE SOLUTION TO THE DEALING PROBLEM

*The Idea is as follows:*

1. A chooses  $KEY_B$  for B.
2. A publishes the chosen encryption keys.
3.  $KEY_B$  are distributed to B using a variant of "the subscription to a public-key protocol", using one root and receipts. As a result B gets a real-key (dummy-key) with probability  $= 1/2$  ( $1/2$ ). B takes keys until he has as many as he needs (say 30) of each subset of keys.
4. Symmetrically B chooses keys for A.
5. The fact that B (A) knows the encryption and the decryption keys of all keys in  $KEY_A$  ( $KEY_B$ ), does not compromise the secret blocking.

*The Improvements to The Dealing Protocol are :*

1. In step 2.b the permutation (P) is chosen at random by A alone.
2. In step 3.b B randomly chooses which *card embeddings* to try to take. He applies the secret blocking to embeddings he decides not to take.
3. In step 3.b B halts the dealing and moves to step 4 at the end of the loop.

As a result of the improvements the following theorem holds:

**Theorem 5:** The "dealing of cards protocol" is correct according to its specification:

- a. If no player cheats, then when a player draws the cards, the following properties hold: *Fairness, Disjointness of Drawn Cards, Security, Verifiability.*
- b. Any case of cheating is detectable.

## 6. THE COMPLETE GAME OF TWO PLAYER MENTAL-POKER

We design a main protocol called *The Game Manager* which organizes the game and links the different steps and we design *Protocols for other game steps* as well. The game steps are:

1. Discarding a card : A moves a card M from *AHAND* to *AUSED* in a secure and checkable way.
2. Opening a card : A moves a card M from *AHAND* to *AOPEN* in a checkable way.
3. Opening a card from *DECK*: first, using protocol 4, A gives a card M to B then B opens M.

It is easy to design these protocols since at the beginning of each of them a new random code of the abstract card sets is used. The order of *ALL* is the interface between steps and we can prove the following theorem:

**Theorem 6:** The two player mental poker game is fair, secure, checkable and a direct simulation of the regular game (using cards) as was specified.

## 7. GENERALIZATION: THE MULTI-PLAYER MENTAL POKER

### 7.1. THE PROBLEM IN MULTI-PLAYER GAME

In the two player game the cards are offered to B from the set  $DECK_A = DECK \cup BSAW$ , and B adds the opened cards he did not previously see to his hand. The disjointness of the cards he takes and cards that have already been seen by A (at any given moment) is a consequence of the fact that the combining both players partial knowledge is the full knowledge of the game, and that *DECK*, *ASAW* and *BSAW* are mutually disjoint while their union is *ALL*. How can we guarantee, however, that B takes cards only from *DECK* and does not get any additional partial information, while  $DECK_{A_i} \cap DECK_B \neq DECK$  in the generalized situation? We must somehow let all the players participate in the dealing and still keep the mutual privacy and security constraints.

**Assumption :** All messages are sent to all players. This is a minimal assumption, because otherwise if even two out of the K players can communicate privately, they can make a coalition and get an advantage over the others just by knowing each others' hands. Also, in order to be able to replay the protocols, we assume that every message is acknowledged by all the players.

**The Changes :** For each player j we define the following sets:  $HAND_j$ ,  $USED_j$  and  $SAW_j$  are respectively the cards in his hand, cards he already used and their union. During the game we keep

$\{ SAW_j \cap SAW_i = \emptyset \text{ for } i \neq j \}$  and  $\{ SAW_j \cap DECK = \emptyset \}$ .

Suppose there are K players. Let B be the K-th player and  $A_i, i=1, \dots, K-1$  all the others. We define

$DECK_{TOB} = \bigcap_{i=1}^{K-1} (DECK_{A_i}) = DECK \cup BSAW$ . This is the set that player B,



who is taking the cards, is allowed to see and to choose cards from. We call these cards Candidate Cards, because such a card is a candidate to be drawn by B, namely if a card  $M \in DECKTOB - BSAW$  then  $M \in DECK$ , and B can take it.

The multi-person dealing protocol is a two-stage protocol: *DECKTOB* generation stage and Card drawing stage. These two stages are two coroutines, each of them has a *current state* and they run concurrently.

## 7.2. THE SOLUTION : A PROTOCOL FOR MULTI-PLAYER DEALING OF CARDS

### Protocol 5 : MULTI-PLAYER DEALING

( $A_i, i=1, \dots, K-1$  deals cards to  $B=A_K$ . They start at stage A.)

*current state of stage A* is : "begin the stage in step 1".

*current state of stage B* is : "begin the stage in step 6".

#### Stage A: DECKTOB GENERATION

The stage starts at its current state:

*step 1* : The  $K-1$  players choose a common random permutation of  $1, \dots, 52 : Q$  (B does not know what  $Q$  is). They embed  $Q$  unambiguously in  $EM(Q)$ , and transfer it to B. (The communication between the  $k-1$  players can be done using a group key, see section 3.)

*step 2* : Every player  $A_i$  chooses his own *private current code* of  $ALL : ALL^i$  each  $A_i \rightarrow B : "Q(f_i(ALL^i))"$ .

(The players do not reveal the cards in the right order, but rather, a random permutation of them.)

*step 3* : For player  $A_i$  let :  $DECK_i = DECK \cup (\cup_{j:j \neq i} \{SAW_j\}) = ALL - SAW_i$ .  $A_i$  embeds each card  $M \in DECK_i$  in  $EM(M)$ .

*step 4* : Each  $A_i$  chooses a private random permutation  $P_i$ .  $A_i \rightarrow B : "P_i(EM(DECK_i))"$

(The cards are offered in order  $P_i$ ,  $A_i$  has to remember this order.)

*step 5* : Opening of cards : B tries to open cards using OBLIVIOUS TRANSFERS, alternately with  $A_i, i=1, \dots, K-1$ . He makes iterations over the embedded cards as in the two player case. During this process B can get the following information :

a. *Factoring of a card embedding* : B gets a card code  $M$  of some of the other players  $A_i$ . He can compute  $f_i(M)$ , but this gives him no idea what  $M$  is because he gets only the place of  $M$  in the permutation  $Q$  which is a random permutation and  $M$  is just a random number.

b. *Getting a candidate card* : During the transfers B realizes that a card is offered to him by *all* the other players, (the same place in  $Q$  was revealed in all  $Q(ALL^i)$ ). This card is either in  $DECK$  or in  $BSAW$  so it belongs to  $DECKTOB$  (it is a candidate card). Suppose B needs  $v$  cards and during the process he gets  $v$  random candidates. Then the players remember *the current state of stage A* and go to Stage B.

end {Stage A}

### Stage B : CARD TAKING

(In this stage only B and one of the players ( $A_i$ ) are playing, but the others get and acknowledge all messages.)

The stage starts at its current state:

*step 6* :  $A_i$  chooses a new code of  $ALL$ :  $ALL^i$ .

$A_i \rightarrow B$ : " $f_1(ALL^i)$ " (This time without permuting the order.)

*step 7* : A embeds each  $M \in DECK_i$  in  $EM(M)$ .

$A_i \rightarrow B$ : " $P_1(EM(DECK_i))$ " ( He uses the *same* permutation  $P_1$  he used in stage A; the permutation of  $DECK_i$  is the *interface* between the two stages.)

*step 8* : B and  $A_i$  use iterations of OBLIVIOUS TRANSFERS in order to let B factor the embeddings. B knows which card in the permutation  $P_1$  is a candidate, using the SECRET BLOCKING he chooses to open only candidates.

*Taking a card* : When B gets all the factors of the embedding of a candidate card, he recovers M and computes  $f_1(M)$ . If  $M \notin BSAW$  he takes it. If B gets all the cards he needs, he stops the process by  $B \rightarrow A_i$ : "stop, I got j cards". If he has already seen some of the candidates, then the players return to stage A.

end {Stage B}

end {protocol 5}

The reduction of the multi-player case to several two-player protocols implies the following:

**Theorem 7:** The protocol for Multi-player Dealing of Cards simulates dealing of cards and has the specified properties of security, verifiability and fairness.

## 8. CONCLUSIONS

We presented cryptoprotocols which can be used with a public-key cryptosystem. The subscription to a public-key and the secret blocking protocols are cryptographic tools, augmenting the power of public-key systems. Developing these tools and their applications and solving the multi-player mental poker game extends our knowledge of the power of cryptographic techniques, the range of applications of these techniques, and the boundaries between the possible applications and the impossible ones. The study of these subjects is one of the main targets of recent cryptographic research.

In designing the protocols, we used a methodology that can be used for designing and proving the correctness of long cryptoprotocols. We observe four main design stages:

1. *The axiomatic stage*: We have two kinds of axioms: a. The underlying mathematics. b. The computational environment: rules of communication, user behavior, etc.

2. *The basic Cryptographic Techniques*: Based on our axioms, we use or construct basic algorithms and cryptotransactions like the RSA system, Oblivious Transfer,

Number-Embedding.

*3. Top-Down Design of the protocol:* The problem at hand is divided into sub-problems (an analogous to modular design of a computer program). For every sub-problem we develop a cryptoprotocol using the basic tools of stage 2. We take care of the security and other specified properties of the sub-problems' protocols, at the same time ensuring the specified properties of the whole process. We use an inference system which includes "security logic" and "process logic". (In our case we prove formal assertions about card sets and users' information and we use the global order of the cards to concatenate steps.)

*4. The Process Protocol:* After stage 3 the process is executed over the communication channels according to the rules of the original process. We also handle additional administrative communication which we ignored when we concentrated on the problem.

This approach of divide and conquer (using the same or other system axioms and proof techniques) will undoubtedly be used in other complex and long cryptoprotocols that will be designed in the future.

## Acknowledgements

I wish to thank my teacher Zvi Galil for his help and encouragement, and Bruce Abramson, Bruce Hillyer, Brian Kwartler and Carmi Weinzig for their useful comments.

## References

1. Angluin D. Lecture notes on the complexity of some problems in number theory. Tech. Rept. 243, Dep. of Computer Science, Yale University, August, 1982.
2. Berlekamp E.R. "Factoring Polynomials over Large Finite Fields." *Mathematics of Computation* 24 (July 1970), 713-735.
3. Blum M. and S. Goldwasser. An Efficient Probabilistic Public-Key Scheme Which Hides All Partial Information. to appear in *Proceedings of Crypto84*, 1984.
4. Blum M. Three Application of the Oblivious Transfer. University of California at Berkely, September, 1981.
5. Blum M. Mental Poker. University of California at Berkely, 1982. to appear
6. Blum M. "How to Exchange (Secret) Keys." *ACM Transactions on Computer Systems* 1, 2 (May 1983), 175-193.
7. Diffie W., and M.E. Hellman. "New Directions in Cryptography." *IEEE Transactions of Information Theory* IT-22 (November 1976), 644-654.
8. Goldwasser, S. and Micali S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. *Proceedings of the 14 Annual ACM Symp. on Theory of Computing, ACM-SIGACT*, May, 1982, pp. 365-377.
9. Knuth D. E.. *The Art of Computer Programming*. Volume 2: *Seminumerical Algorithms*. Addison-Wesly, Reading, Massachusetts, 1981.
10. Lipton R. How to Cheat at Mental Poker. *Proceeding of the AMS short course on Cryptography*, AMS, January, 1981.
11. Niven I. and Zuckerman H.S.. *An Introduction to the Theory of Numbers*. Wiley, New York, 1981.
12. Rabin M. Digitalized Signatures and Public-key Functions as Intractable as Factorization. Tech. Rept. LCS/TR-212, MIT, January", 1979.
13. Rabin M. Probabilistic Algorithms in Finite Fields. Tech. Rept. LCS/TR-213, MIT, January, 1979.
14. Rackoff C., S. Micali and M. Fischer. A Secure Protocol for the Oblivious Transfer. *Eurocrypt 84*, La Sorbonne, Paris, April, 1984.
15. Rivest R. , Shamir A., Adleman L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM* 21, 2 (February 1978), 120-126.

16. Shamir A. "How to Share a Secret." *Communication of the ACM* 22, 11 (November 1979), 612-613.
17. Shamir A., Rivest R. Adleman L. Mental Poker. In *Mathematical Gardner*, Klarner D. E., Ed., Wadsworth Intrntl, 1981, pp. 37-43.
18. Solovay R., and Strassen V. "A Fast Monte-Carlo Test of Primality." *SIAM Journal on Computing* 6 (March 1977), 84-85.
19. Yung M. K-Player Mental Poker. Master Th., Tel-Aviv University, March 1982.

# POKER PROTOCOLS

Steven Fortune  
Michael Merritt

AT&T Bell Laboratories  
600 Mountain Ave  
Murray Hill, NJ 07974

## 1. INTRODUCTION

The situation is quite serious. After four years of research, there has been no satisfactory way for a group of card sharks to play poker over the phone. Until now. In this paper, we present a new method for playing 'mental poker,' discuss its significance, and mention some of the further questions it raises. Ante up.

The rules for mental poker are just like regular poker, except that players communicate over the phone, and there are no physical cards. The hard part of mental poker is dealing the cards. Hands must be random and disjoint, and players should not be able to claim to have any cards but those dealt (a sleeve will hold as many 'virtual cards' as angels will fit on the head of a pin).

Playing mental poker is a difficult problem for a number of reasons. The foremost reason is that it is impossible, a result due to Shamir, Rivest and Adleman.<sup>[1]</sup> Of course, this is an information-theoretic result, and the same reference presents a method for playing mental poker that relies on the difficulty of inverting certain cryptographic transformations. Unfortunately, a cryptographic flaw allows players to determine the color of each other's cards.<sup>[2]</sup> This set the stage for a new implementation devised by Goldwasser and Micali, which was proven to hide all partial information (up to an explicit cryptographic assumption).<sup>[3]</sup> Unfortunately, this implementation works only for two players, which is a very restricted kind of poker. Next, Barany and Furedi devised a protocol that permits three or more players to play poker,<sup>[4]</sup> but only if players are not permitted to form coalitions. If two players conspire, they can learn the contents of everyone else's hands. The following section discusses this history of mental poker in more detail, outlining the key ideas, contributions and limitations of this earlier work.

This paper presents a new way of playing mental poker. Unlike earlier solutions, it is secure against coalitions, permits any number of players, and uses inexpensive, highly secure cryptographic techniques. The protocol does require the participation of a trusted party to shuffle the cards. However, thereafter the trusted party does not participate in the protocol. The protocol can be easily adapted to play almost

all types of poker known to the authors.

Of course, poker is a metaphor for any system in which users should have only partial information about the dynamic allocation of resources. Beyond this, the poker protocol presented here takes on a broader significance because of the simple tools used in its implementation.

## 2. THE HISTORY OF MENTAL POKER

### 2.1 A Protocol based on Commutativity

To our knowledge, the first attempt to play mental poker was made by Niels Bohr during a skiing vacation near Oberaudorf in 1933. At Bohr's instigation, he, his son Christian, Felix Bloch, Carl Friedrich, and Werner Heisenberg attempted to play poker without cards, each player calling out the contents of his nonexistent hand. Heisenberg reports "The attempt was made, but did not lead to a successful game." Apparently, the players did not pursue the problem further, finding adequate challenge in other research areas.<sup>[5]</sup>

Much later, the problem was independently posed by Robert Floyd. This led to the first formal results on mental poker, in a 1979 research report by Adi Shamir, Ronald Rivest, and Leonard Adleman, later published in *The Mathematical Gardner*<sup>[1]</sup>. In the spirit of modern cryptography, they began by proving the problem impossible to solve. The argument is very simple: if Alice and Bob are playing poker, either Bob can claim to have a straight-flush every time, or Alice has enough information to determine Bob's hand. Having established the problem as adequately challenging, Shamir, *et al* proceeded to solve it. Their solution circumvents the impossibility result by exploiting 'hidden' information. Alice knows Bob's hand in the sense that it is the unique solution to a computational problem, but finding that solution is beyond her capabilities. Verifying the solution, once Bob divulges his hand, is easy.

More specifically, this first poker protocol utilizes the power of *commutative* cryptosystems. Let  $E_A$  and  $D_A$  be Alice's encryption and decryption functions, respectively, and let  $E_B$  and  $D_B$  be Bob's. Suppose that  $E_A(D_B(x)) = D_B(E_A(x))$  and  $E_B(D_A(x)) = D_A(E_B(x))$  for all messages  $x$ . Then Alice and Bob play a hand of poker as follows (it is a simple exercise to extend this protocol to three or more players).

Let the deck of cards be any encoding of the set  $\{1, \dots, 52\}$  appropriate for the cryptosystem. Alice encrypts each card in the deck separately, randomly orders the resulting set  $\{E_A(1), \dots, E_A(52)\}$ , and sends it to Bob.

Bob chooses five encrypted cards at random, say  $\{E_A(18), E_A(24), E_A(27), E_A(31), E_A(39)\}$ , and sends them back to Alice, who now knows her hand is  $\{18, 24, 27, 31, 39\}$ .

Next, Bob chooses five different encrypted cards, say  $\{E_A(3), E_A(12), E_A(15), E_A(35), E_A(41)\}$ , encrypts them, and sends the randomly ordered set  $\{E_B E_A(3), E_B E_A(12), E_B E_A(15), E_B E_A(35), E_B E_A(41)\}$  back to Alice.

Alice decrypts each element of the set, and sends the resulting set,  $\{E_B(3), E_B(12), E_B(15), E_B(35),$

$E_B(41)\}$ , back to Bob.

Bob decrypts the set to get his hand,  $\{3,12,15,35,41\}$ .

Once the hand has been played, Alice and Bob exchange their encryption keys and verify that each played fairly (until this point, both players could claim to have any hand they like, so long as it did not happen to intersect their opponent's hand). Both Alice and Bob's encryption keys must be uniquely determined by the messages sent during the deal. Otherwise, Alice or Bob could divulge the key which decodes the best hand. It is also important that the encryption function hide *all* of the message—leaking even a single bit about a card (such as its color) can make a significant difference.

Shamir, Rivest and Adleman suggest a particular commutative cryptosystem for implementing their protocol. It is based on modular exponentiation. Alice and Bob agree on a large odd prime number  $n$ , and separately choose secret keys  $k=A$  or  $k=B$ , where  $\gcd(A, n-1)=\gcd(B, n-1)=1$ . Then  $E_k(x)=x^k \pmod n$  and  $D_k(x)=x^{k^{-1}} \pmod n$ , where  $kk^{-1} \equiv 1 \pmod{n-1}$ .

## 2.2 These Cards are Marked!

Shortly after this protocol and implementation appeared in a technical report, Lipton observed that this implementation leaks at least a bit of information<sup>[2]</sup>. This is because exponentiation modulo  $n$  preserves *quadratic residues*. A number  $x$  is a quadratic residue modulo  $n$  provided  $x \equiv y^2 \pmod n$  for some  $y$ . Otherwise,  $x$  is a quadratic nonresidue. Half of the integers are quadratic residues modulo  $n$ , for  $n$  a large prime. For such  $n$  it is easy to determine whether a number is a quadratic residue. Finally, since  $k$  must be odd,  $x^k \pmod n$  is a quadratic residue if and only if  $x$  is. Thus, knowing which cards are quadratic residues, and comparing these against the encrypted cards Alice sends him, Bob has about a bit of information per card.

Of course, the cards could be encoded originally so that they are all quadratic residues (or all quadratic nonresidues). Lipton discusses this and other suggestions for strengthening the cryptosystem, but notes that there is still no guarantee that the result is secure. Indeed, the indication is that bits may still leak.

## 2.3 A Provably Secure Two-Person Game

Some three years after Lipton's observation, Goldwasser and Micali published a new protocol for playing mental poker<sup>[3]</sup>. A major achievement of their work was a proof that discovering a single bit of an opponent's hand was equivalent to solving an apparently intractable problem (factoring, index finding or deciding quadratic residuosity with respect to composite moduli). Unfortunately, their protocol works only for two players.

The details of their protocol are beyond the scope of this survey. What follows is a very simplified description, intended to explain why it works for only two players.

Our friends Alice and Bob will play again. Alice shuffles a deck of cards, encrypts it using a function  $A$ , and sends the encrypted deck to Bob. Similarly, Bob shuffles a deck, encrypts it using  $B$ , and sends it to Alice. Micali and Goldwasser show how Bob can ask Alice to decrypt one card of her deck, without Alice knowing which card she has decrypted. This technique is the crux of their protocol and uses some clever computational number theory; essentially Bob asks a question about every encrypted card, but for



only one card does Bob gain enough information to decrypt the card. (There is a later verification step that ensures that Bob didn't decrypt more than one card.)

So how is the game played? Suppose Bob has drawn card  $x$  from Alice's deck. He then removes  $B(x)$  from his own encrypted deck, so when it is Alice's turn to draw, she can't choose  $B(x)$ . Similarly, any card drawn by Alice is removed from her deck, so Bob can't draw it. Neither player knows what cards have been removed from the opponent's deck, since the decks are encrypted.

Now it is clear why the protocol works for only two players. Alice draws from Bob's deck, which does not contain Bob's hand, and Bob draws from Alice's deck, which does not contain Alice's hand. If Charles wants to play, which deck does he choose from? If he chooses from Bob's deck, he might get a card Alice has, and if he chooses from Alice's deck, he might get a card Bob has.

## 2.4 Three or More Players

What happens with three or more players? Suppose Charles wishes to play with Alice and Bob. To keep Alice from falsely claiming to have a straight-flush, Bob and Charles must together have enough information to determine her hand. But neither alone need have this information. And as long as they cannot share information, they remain ignorant of Alice's hand. Thus, the impossibility argument of Shamir, *et al*, does not work with three or more players.

This observation was made by Barany and Furedi<sup>[4]</sup>, who also present a simple protocol for mental poker for three or more players. Let's see how Alice, Bob and Charles can use this protocol to play poker. Initially, each player chooses a random permutation of the deck,  $A$ ,  $B$  and  $C$ , respectively.

Next, Bob and Charles send  $B$  and  $C$  to Alice. They do this secretly, so that they remain ignorant of each other's permutation.

Now Alice secretly sends  $BA^{-1}$  to Charles, and  $CA^{-1}$  to Bob.

Now the cards are ready to be dealt. Let's jump ahead to a point where Alice, Bob and Charles already have some cards, the sets  $H_A$ ,  $H_B$  and  $H_C$ , respectively, and see how new cards will be dealt. (Initially, these are the empty sets). We assume that the hands are disjoint so far.

At this stage, each player has the following information.

Alice knows  $A$ ,  $B$ ,  $C$ ,  $H_A$ .

Bob knows  $B$ ,  $CA^{-1}$ ,  $H_B$ ,  $A(H_A)$ ,  $A(H_B)$ ,  $A(H_C)$ .

Charles knows  $C$ ,  $BA^{-1}$ ,  $H_C$ ,  $A(H_A)$ ,  $A(H_B)$ ,  $A(H_C)$ .

Suppose Bob wants a new card. He gets it from Charles as follows. Charles chooses a number  $x$  not in  $A(H_A)$ ,  $A(H_B)$  or  $A(H_C)$ , and sends  $BA^{-1}(x)$  to Bob, from which Bob computes his card,  $y = A^{-1}(x)$ . Bob adds  $y$  to  $H_B$ , and both Bob and Charles add  $x$  to  $A(H_B)$ . Charles gets a card from Bob in a similar way.

When Alice wants a new card, she gets it from either Bob or Charles, say Bob. Bob chooses a number  $x$  not in  $A(H_A)$ ,  $A(H_B)$  or  $A(H_C)$ , and sends  $x$  to both Alice and Charles. Alice adds  $y = A^{-1}(x)$  to  $H_A$ , and Bob and Charles add  $x$  to  $A(H_A)$ .

Alice plays a special role in this protocol. Thinking of her permutation as a shuffled deck of cards, everyone but Alice knows which cards in this shuffled deck they each hold. For instance, Bob and Charles may know Alice has the 3<sup>rd</sup> and 4<sup>th</sup> cards in the deck, Bob has the 34<sup>th</sup> and 51<sup>st</sup> and Charles has the 22<sup>nd</sup>. By picking cards in  $A$  that have not yet been dealt, Bob and Charles can keep all hands disjoint. Because only Alice knows how the cards are ordered by  $A$ , the cards Bob and Charles pick will be randomly chosen.

The assumption that players will not collude is crucial to the protocol. If any two players share their knowledge, they learn not only each other's hands, but their opponent's hands, as well. No one would bet real money under these conditions. In Section 3, we show how Alice's special role can be played by a trusted party during an initialization stage, in such a way that players who collude learn only the contents of each other's hands.

## 2.5 Other Poker Protocols

A protocol based on ideas similar to the Goldwasser-Micali protocol was independently proposed by Yung.<sup>[6]</sup> Yung's protocol improves on the Goldwasser-Micali protocol by allowing more than two poker players. Unfortunately, his protocol is quite complex, and like the Barany-Furedi protocol, it is not secure against collusion: two players collectively have enough information to deduce other players' hands as well.

One approach to prevention of collusion was suggested by Fich and Goldwasser.<sup>[7]</sup> The Fich-Goldwasser protocol is based on the Barany-Furedi protocol; the novelty is that it is not possible to transmit secret information in the messages of the protocol itself. If all communication between participants is restricted to protocol messages, then collusion is impossible. Of course, if there is some secret channel between players, say a surreptitious telephone line, then the protocol suffers the same defect as the Barany-Furedi protocol, and two colluding players can learn all hands.

## 3. PRACTICAL MENTAL POKER

### 3.1 Introduction

Can we construct a poker protocol for three or more players that is secure against collusion? We now do so. Our protocol uses the "distributed-information" technique of Barany and Furedi. In addition, it uses one-way functions to authenticate information, and the services of a "Card Salesman" as a trusted participant.

The Card Salesman participates in the protocol at the beginning of play. He receives a small amount of secret information from each player, then publicizes information that allows play to proceed. The Card Salesman must be a trusted participant, for he has enough information to discover all players' hands. The advantage of a Card Salesman over a trusted dealer (in which case the poker problem is trivial) is that the Card Salesman need only participate at the beginning of play.

In many ways the role of Card Salesman is analogous to the manufacturer of a deck of cards. Serious poker players insist on beginning any poker game with a new deck of cards, in a box still sealed by some trusted manufacturer. The players then have some assurance that the cards are not marked. Both in the case of the card manufacturer and of the Card Salesman, the cost of the trusted participant is small.

The poker protocol also requires the use of one-way functions. A one-way function  $f$  is a function that is easy to compute and hard to invert. One-way functions are valuable authentication tools. For instance, computer systems often store encryptions of passwords, rather than cleartext passwords.<sup>[8]</sup> Although there is no proof that there is any easy-to-compute function that is hard to invert (since such a proof would imply  $P \neq NP$ ), practical one-way functions are easy to construct. This is because there is no need to construct the inverse of the one-way function, quite in contrast to the case of public-key or private-key encryption methods. We assume the existence of a one-way function  $f$  that is either one-one, or if not one-one then, given  $z=f(y)$ , it is computationally hard to find any  $x$  so that  $f(x)=z$ . Further discussion of one-way functions can be found in the references<sup>[8] [9]</sup>.

The poker protocol has the following general format. At initialization, each player chooses some secret information. At later stages, each player makes a move that either is based on a random choice, or is completely determined by the secret information initially chosen. The fairness of the protocol depends upon the player abiding by the initially chosen secret information. To convince other players that he is not cheating, at the beginning of play each player broadcasts his secret information, encrypted by a one-way function. At the end of the game, each player broadcasts his secret information, unencrypted. Then all players can check that all other players followed the protocol. Note that since all messages besides secret messages are broadcast to all players, the information needed to check other players behavior is available. Also, since one-way functions are hard to invert, after play is over, a player cannot broadcast secret information different from what he was using during play.

As long as the Card Salesman and at least one player play fairly, no group of colluding players can gain an unfair advantage over other players. The proof of this assumes that the one-way function cannot be inverted. Actually, a stronger assumption is necessary, that no statistical information at all about the preimage of a function value can be inferred. This assumption becomes apparent in the analysis of the protocol, as we ignore the fact that the encrypted secret information has been published.

### 3.2 The protocol

The poker protocol assumes that two network services are available: the ability to send secret messages between pairs of players, and the ability to broadcast a message to all players. Actually, secret messages are only sent at the initialization of the protocol, and then only to the Card Salesman. All other messages are broadcast to all players. We assume that the network reliably provides these two services.

The duty of the Card Salesman is to choose a random permutation  $\pi$  that encodes players hands. Suppose Alice has a hand  $H_A$ . Of course  $H_A$  is not known to other players, but  $\pi(H_A)$  will be public knowledge. No player has information about  $\pi$  beyond the value of  $\pi$  on his hand. To draw a new card, the player must choose some  $y=\pi(x)$  not in any other player's hand. This is possible since the player knows the  $\pi$ -encoded form of the other players' hands. The poker protocol then reveals to the player  $x=\pi^{-1}(y)$  without letting any other player know  $x$ .

So how does a player, say Charles, draw a card? Before the game starts, Alice, Bob, and Charles each choose a random permutation  $\alpha$ ,  $\beta$ , and  $\gamma$ , respectively, and transmit their permutation secretly to the Card Salesman. The Card Salesman computes the product  $\Delta = \alpha^{-1}\beta^{-1}\gamma^{-1}\pi^{-1}$  and broadcasts it to all players. Now suppose Charles wishes to draw a card. He randomly chooses some  $y = \pi(x)$  not in any other player's hand, and broadcasts  $y$  and  $\Delta(y)$ . Alice now computes and broadcasts  $\alpha(\Delta(y))$ . Bob computes and broadcasts  $\beta(\alpha(\Delta(y)))$ . Finally, Charles computes  $\gamma(\beta(\alpha(\Delta(y)))) = x$ , and of course does not broadcast it. Note that the same permutations  $\alpha$ ,  $\beta$ ,  $\gamma$  can be used the next time Charles draws a card. However, permutations  $\alpha$ ,  $\beta$ ,  $\gamma$  cannot be used to draw cards for a different player.

We now describe the complete poker protocol for three players, Alice, Bob, and Charles. The generalization to more than three players is straightforward.

Initialization:

1. The Card Salesman randomly chooses a permutation  $\pi$ .
2. Alice chooses three permutations  $\alpha_A, \alpha_B, \alpha_C$ . Similarly, Bob and Charles each choose three permutations  $\beta_A, \beta_B, \beta_C$  and  $\gamma_A, \gamma_B, \gamma_C$ . All permutations are transmitted secretly to the Card Salesman, and their encryptions using the one-way function are broadcast.
3. The Card Salesman computes and broadcasts the products  $\Delta_A = \beta_A^{-1}\gamma_A^{-1}\alpha_A^{-1}\pi^{-1}$ ,  $\Delta_B = \gamma_B^{-1}\alpha_B^{-1}\beta_B^{-1}\pi^{-1}$ , and  $\Delta_C = \alpha_C^{-1}\beta_C^{-1}\gamma_C^{-1}\pi^{-1}$ .

For Charles to draw a card:

1. Charles randomly chooses  $y = \pi(x)$  not in any player's hand and broadcasts  $y$  and  $\Delta_C(y)$ .
2. Alice broadcasts  $\alpha_C(\Delta_C(y))$ .
3. Bob broadcasts  $\beta_C(\alpha_C(\Delta_C(y)))$ .
4. Charles computes  $x = \gamma_C(\beta_C(\alpha_C(\Delta_C(y))))$ .
5. All players record that Charles has  $y = \pi(x)$  in his hand.

For Alice to draw a card, Alice publishes  $y$  and  $\Delta_A(y)$ , then Bob and Charles broadcast  $\beta_A(\Delta_A(y))$  and  $\gamma_A(\beta_A(\Delta_A(y)))$ , respectively. Similarly, for Bob to draw a card, Bob publishes  $y$  and  $\Delta_B(y)$ , then Charles and Alice broadcast  $\gamma_B(\Delta_B(y))$  and  $\alpha_B(\gamma_B(\Delta_B(y)))$ , respectively.

End of play: Each player publishes his permutations, and checks that every other player played fairly. Then the debts are settled.

### 3.3 Correctness

Is it possible to play poker with this protocol? Certainly. It is clear that one round of the protocol allows a player to draw a card not in anyone else's hand. The more important question is: is anyone willing to play poker with this protocol? I.e., is it possible to guarantee absence of cheating? It is clear that before any cards are dealt, and in the absence of collusions among the players,  $\pi$  is random, so a player has no better strategy for drawing cards than random choice. But what happens in the presence of collusions, and after cards are dealt? We show that as long as the Card Salesman and at least one player play fairly, that is, they choose their permutations randomly with uniform distribution and conceal them from other players, then no other player or group of colluding players can gain any information on cards

not in their own hands.

How can we analyze the protocol? We wish to determine the probability distribution of  $\pi$ , given the information a player or group of players have at some point during play. We know that originally  $\pi$  was chosen randomly by the Card Salesman. However, the Card Salesman publishes the permutations  $\Delta$ , and as play proceeds information is made public about each player's private permutations. To analyze the probability distribution of  $\pi$ , even in the presence of these additional constraints, we set up a state space that captures the possible values for  $\pi$  and also all of the permutations initially chosen by the players. Note that the state space does not capture the later random choices made by the players. The state space has a probability distribution, given by the probabilities with which the players choose their permutation. A player's knowledge is modelled as a subset of the possible states, specifically, all the states satisfying the player's knowledge. We determine the conditional probability distribution for  $\pi$ , given that the state must satisfy the knowledge of a player or group of players.

The analysis is given in terms of three players Alice, Bob, and Charles, assuming that Alice plays fairly, and Bob and Charles possibly collude. The generalization to more than three players is straightforward.

The state space has variables  $A_A, A_B, \dots, \Gamma_C$  and  $\Pi$ . Each of these variables has as possible values a permutation on  $1 \dots 52$ . A state is a particular set of values for the variables,  $A_A = \alpha_A, A_B = \alpha_B, \dots, \Gamma_C = \gamma_C$ , and  $\Pi = \pi$ . We use upper case letters for variables and lower case letters for particular values. Knowledge is denoted by a set of equations involving the variables; the set of equations specifies all states satisfying the equations.

What is known, publicly and privately? We summarize it as follows.

Alice's knowledge,  $K_A: A_A = \alpha_A, A_B = \alpha_B$ , and  $A_C = \alpha_C$

Bob's knowledge,  $K_B: B_A = \beta_A, B_B = \beta_B$ , and  $B_C = \beta_C$

Charles's knowledge,  $K_C: \Gamma_A = \gamma_A, \Gamma_B = \gamma_B$ , and  $\Gamma_C = \gamma_C$

Public Knowledge,  $P$  (consisting of all messages that have been broadcast):

$$\Delta_A = B_A^{-1} \Gamma_A^{-1} A_A^{-1} \Pi^{-1}$$

$$\Delta_B = \Gamma_B^{-1} A_B^{-1} B_B^{-1} \Pi^{-1}$$

$$\Delta_C = A_C^{-1} B_C^{-1} \Gamma_C^{-1} \Pi^{-1}.$$

For each  $x$  in  $\Pi(H_A): B_A(\Delta_A(x)) = \beta_A(\Delta_A(x))$  and  $\Gamma_A(\beta_A(\Delta_A(x))) = \gamma_A(\beta_A(\Delta_A(x)))$

For each  $x$  in  $\Pi(H_B): \Gamma_B(\Delta_B(x)) = \gamma_B(\Delta_B(x))$  and  $A_B(\gamma_B(\Delta_B(x))) = \alpha_B(\gamma_B(\Delta_B(x)))$

For each  $x$  in  $\Pi(H_C): A_C(\Delta_C(x)) = \alpha_C(\Delta_C(x))$  and  $B_C(\alpha_C(\Delta_C(x))) = \beta_C(\alpha_C(\Delta_C(x)))$

Lemma. Let  $k = |H_B \cup H_C|$ . In any probability distribution where  $\Pi, A_A, A_B$ , and  $A_C$  are each uniformly distributed and independent of all other variables, then for any  $x$  not in  $H_B \cup H_C$ ,  $y$  not in  $\Pi(H_B) \cup \Pi(H_C)$ ,  $\Pr(\Pi(x) = y | K_B K_C P) = 1/(52 - k)$ .

Proof: We need to analyze the set of states consistent with the partial information  $K_B, K_C$ , and  $P$ ; in particular, we need to know the possible values of  $\Pi$ . Say that  $\pi$  behaves correctly if for all  $x$  in  $H_B$ ,  $\pi(x) = \Delta_B^{-1}(c_B^{-1}(a_B^{-1}(b_B^{-1}(x))))$  and for all  $x$  in  $H_C$ ,  $\pi(x) = \Delta_C^{-1}(a_C^{-1}(b_C^{-1}(c_C^{-1}(x))))$  (that is,  $\pi$  has the correct value on  $H_B$  and  $H_C$ ). Note that all these values are part of the partial information  $K_B, K_C$ , and  $P$ . We claim that if  $\pi$  behaves correctly, then there is a state with  $\Pi = \pi$  satisfying the equations in  $K_A, K_B, K_C$ , and  $P$ , and if  $\pi$  does not behave correctly, then there is no state with  $\Pi = \pi$  satisfying the equations.

Furthermore, once  $\pi$  is chosen, since the variables  $B_A, B_B, B_C, \Gamma_A, \Gamma_B, \Gamma_C$  all have values fixed at  $\beta_A, \beta_B, \beta_C, \gamma_A, \gamma_B, \gamma_C$ , respectively, the values of  $A_A, A_B$ , and  $A_C$  are uniquely determined. This follows by just examining the equations. Since each of  $\Pi, A_A, A_B$ , and  $A_C$  are uniformly distributed and independent, each state arising from a choice of  $\pi$  is equally likely. What proportion of the choices of  $\pi$  satisfy  $\pi(x)=y$ ? Clearly,  $1/(52-k)$ .

QED

So can Bob and Charles collude? They can of course tell each other their own cards. But by the Lemma, they can gain no information on cards that are not in their own hands. Thus, they cannot infer the cards in Alice's hand, nor can they do better than a random choice in choosing a card to draw. Of course, this assumes that Alice plays fairly. But it is always in Alice's interest to play fairly, since she is then protected from cheating by other players.

As a technical note, we observe that the Card Salesman could choose the players' private permutations in addition to the permutation  $\pi$ . Then the Card Salesman would broadcast the permutations  $\Delta$ , as before, and secretly communicate to each player his private permutations. It would be possible to implement this protocol, but it is probably easier to have all secret communication directed to the Card Salesman, as in the protocol presented.

### 3.4 The Many Flavors of Poker

While purists complain, the name poker applies to a wide variety of sins. Simple games like Seven-card Stud or Five-card Draw can be played using the protocol above directly. Cards can be drawn from the deck, discarded and turned face-up. More complex games are easily accommodated, but there are a few we cannot play.

One author plays an abomination called Indian, in which each player is dealt a single card, face-down. Everyone's card is then shown to every other player, but remains unknown to its holder. After a round of betting, with opportunities to fold, the high card wins. A simple adaptation of our protocol allows one to play this game, and in general to identify a card to any subset of players in a way that can be verified later.

Other games involve passing cards from one player to another. This is easy to do once, just by passing the encrypted card, which is then decoded for the new recipient. But in the game Anaconda, cards are passed more than once, and this implementation allows one to determine that a card one passed earlier is (or is not) being passed in subsequent rounds. A secret and subsequently authenticable message could be used to pass a card and avoid this problem, but the game is hardly worth the added effort.

One thing we cannot do is return one or more cards to the deck and reshuffle it. We don't know any poker games which require this, but they probably exist. In fact, we need to buy a new deck from the Card Salesman for every hand of poker we play. Luckily, decks are cheap, and it is easy to buy lots of them during a single initialization.

## 4. DISCUSSION

The protocol presented here is practical enough to implement and run in real time, even on networks of

small home computers. With the proper choice of one-way function (and Card Salesman), it is actually secure enough to use. This alone is a great improvement over previous solutions.

As mentioned previously, the protocol is also significant because the tools used to play poker are very simple—products of permutations to hide information, one-way functions to permit verification. This point raises several interesting questions. What other, more powerful capabilities can be implemented using such simple tools? A notable example is the probabilistic signature scheme due to Rabin,<sup>[9]</sup> which uses only one-way functions. Is there a way of characterizing the power of these tools and of establishing their limitations?

The poker problem is trivial if there is a trusted Dealer to distribute cards. The Card Salesman is not nearly as expensive as a Dealer (we claim), but we'd really like to get rid of him completely. The protocol here requires the Card Salesman to choose the random  $\Pi$  and to compute and broadcast the permutations  $\Delta$ . Thus we would like to find a way of broadcasting a product of  $n$  secret permutations without divulging any factors. The RSA, and other commutative encryption schemes could be used to do this. Can it be done with only one-way functions and/or private-key systems?

Poker differs from some applications, in that authentication can be carried out in a single stage, after the remainder of the protocol. Other protocols, such as the secret-ballot elections of Chaum<sup>[10]</sup> and Merritt,<sup>[11]</sup> or electronic funds transfers, require authentication during the protocol itself. How important is this distinction, and what tools are required to implement the two types?

#### 4.1 Acknowledgements

We would like to thank Barbara Smith-Thomas and Joan Feigenbaum for their helpful comments on this paper.

## References

1. Shamir, A., Rivest, R., Adleman, L., "Mental Poker", in *Mathematical Gardner*, D.E. Klarner, ed., Wadsworth International, 1981, pp. 37-43.
2. Lipton, R. "How to Cheat at Mental Poker," Proceedings of the AMS Short Course in Cryptography, 1981.
3. Goldwasser, S., Micali, S., "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information," Proceedings of the 14th STOC, pp. 365-377, 1982.
4. Barany, I., Furedi, Z., "Mental Poker with Three or More Players," Technical Report, Mathematical Institute of the Hungarian Academy of Sciences (1983).
5. Heisenberg, W., "Physics and Beyond", translated by Arnold Pomerans, Harper & Row, 1971, p. 139.
6. Yung, M. M., "Cryptoprotocols: Subscription to a Public Key, The Secret Blocking and the Multi-Player Mental Poker Game", manuscript, 1984.
7. Fich, F., Goldwasser, S., "Sending Messages without Hidden Information and Applications to Multiperson Games," manuscript, 1984.
8. Evans, A., Kantrowitz, W., Weiss, E., "A User Authentication Scheme Not Requiring Secrecy in the Computer." CACM (August 1974) Vol. 17(8), pp. 437-442.
9. Rabin, M., "Digitalized Signatures," in Foundations of Secure Computation, DeMillo, Dobkin and Lipton, eds., Academic Press, 1978, pp. 155-168.
10. Chaum, D., "Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms," CACM (February 1981) Vol. 24(2), pp. 84-88.
11. Merritt, M., "Cryptographic Protocols," Ph.D. thesis, Georgia Institute of Technology (1983) GIT-ICS-83/06.



# A "PARADOXICAL" SOLUTION TO THE SIGNATURE PROBLEM\*

Shafi Goldwasser\*\*

Silvio Micali\*\*

Ronald L. Rivest\*\*

## Brief Abstract<sup>(1)</sup>

We present a general signature scheme which uses any pair of trap-door permutations  $(f_0, f_1)$  for which it is infeasible to find any  $x, y$  with  $f_0(x) = f_1(y)$ . The scheme possesses the novel property of being robust against an adaptive chosen message attack: no adversary who first asks for and then receives signatures for messages of his choice (which may depend on previous signatures seen) can later forge the signature of even a single additional message.

For a specific instance of our general scheme, we prove that

(1) forging signatures is provably equivalent to factoring (i.e., factoring is polynomial-time reducible to forging signatures, and vice versa)

while

(2) forging an additional signature, after an adaptive chosen message attack is still equivalent to factoring.

Such a scheme is "paradoxical" since the above two properties were believed (and even "proven" in the folklore) to be contradictory.

The new scheme is potentially practical: signing and verifying signatures are reasonably fast, and signatures are not too long.

\* This research was supported by NSF grant MCS-80-06938, and IBM/MIT Faculty Development Award, and DARPA contract N00014-85-K-0125.

\*\* MIT Laboratory for Computer Science, Cambridge, MA 02139

<sup>(1)</sup> A fuller version of this paper can be found in the Proceedings of the 25th Annual Symposium on Foundations of Computer Science, Singer Island, Florida, October, 1984, pages 441-448.

# SEQUENCE COMPLEXITY AS A TEST FOR CRYPTOGRAPHIC SYSTEMS

A. K. Leung & S. E. Tavares

Department of Electrical Engineering  
Queen's University, Kingston, Ontario, Canada

## ABSTRACT

The complexity of a finite sequence as defined by Lempel and Ziv is advocated as the basis of a test for cryptographic algorithms. Assuming binary data and block enciphering, it is claimed that the difference (exclusive OR sum) between the plaintext vector and the corresponding ciphertext vector should have high complexity, with very high probability. We may refer to this as plaintext/ciphertext complexity. Similarly, we can estimate an "avalanche" or ciphertext/ciphertext complexity. This is determined by changing the plaintext by one bit and computing the complexity of the difference of the corresponding ciphertexts. These ciphertext vectors should appear to be statistically independent and thus their difference should have high complexity with very high probability. The distribution of complexity of randomly selected binary blocks of the same length is used as a reference. If the distribution of complexity generated by the cryptographic algorithm matches well with the reference distribution, the algorithm passes the complexity test. For demonstration, the test is applied to modulo multiplication and to successive rounds (iterations) of the DES encryption algorithm. For DES, the plaintext/ciphertext complexity test is satisfied by the second round, but the avalanche complexity test takes four to five rounds before a good fit is obtained.

## INTRODUCTION

A block enciphering algorithm may be regarded as a reversible transformation which maps binary  $n$ -vectors into binary  $n$ -vectors, for a given key. In modern cryptography it is usually assumed that the cryptographic algorithm is known and only the key is kept secret. In principle, a cryptographic scheme can always be broken by an exhaustive key search. However, if the key set is large, such a search becomes computationally infeasible. On the other hand, if the cryptographic algorithm is not well designed, the key may be discovered with high probability by searching a much smaller set. Thus there is a need to develop statistical tests to reveal such weaknesses. A recent and interesting test is the complexity test. We will discuss some properties of complexity in this paper and apply the test to modulo multiplication and the DES encryption algorithm.

## THE COMPLEXITY CRITERION

Lempel and Ziv [1] introduced the idea of the complexity of a finite sequence and developed several of its important properties. Fischer [2,3] recognised the application of complexity to cryptographic algorithms. Spencer and Tavares [4] applied the complexity test to a layered broadcast cryptographic system and found it to be quite sensitive. Intuitively, the complexity of a sequence is a measure of the rate at which new patterns emerge as we move along the sequence. Starting at one end, say the left, we put a marker whenever a new sequence appears. The complexity is the number of distinct patterns which have been identified. To illustrate, consider the sixteen bit sequence

$$\underline{X} = 1001101110000111.$$

Inserting a marker after each new pattern, we have

$$\underline{X} = 1|0|01|101|1100|0011|$$

and thus  $\underline{X}$  has a complexity of 6. Lempel and Ziv showed that, in the limit, almost all binary sequences of length  $n$  have complexity exceeding  $n/\log n$ . Thus for sequences of length  $n$ , the expression  $C_n = n/\log n$ , may be regarded as a threshold of complexity. If we compute the complexity of a large number of randomly selected binary sequences of length  $n$ , we can determine an IDEAL distribution of complexity as shown in Fig. 1 for sequences of length  $n = 64$ . The above sequences

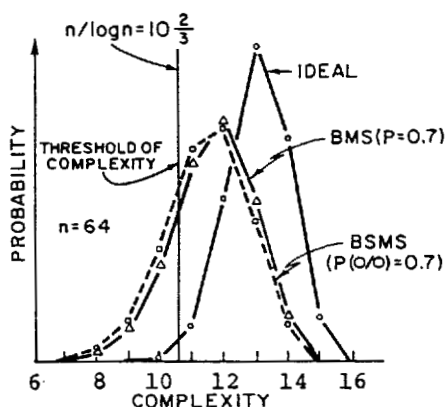


Fig. 1: Distribution of sequence complexity for 64-bit sequences from a selection of binary sources. The IDEAL curve is derived from a Binary Memoryless Source (BMS) with equiprobable symbols. The curve labelled BMS is based on a BMS with  $p(0)=0.7$  and the dashed curve is based on a Binary Symmetric Markov Source (BSMS) with  $p(0/0)=p(1/1)=0.7$ .

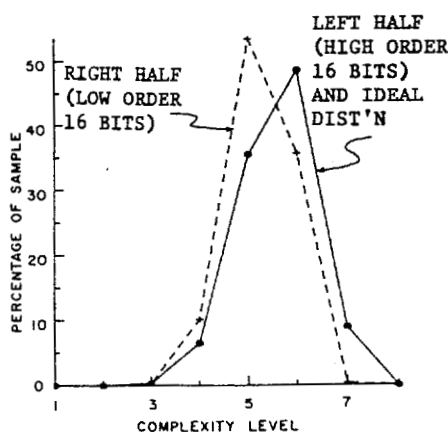


Fig. 3: Distribution of complexity for right half (low order 16 bits) and left half (high order 16 bits) for 32-bit multiplications, modulo  $2^{32}$ .

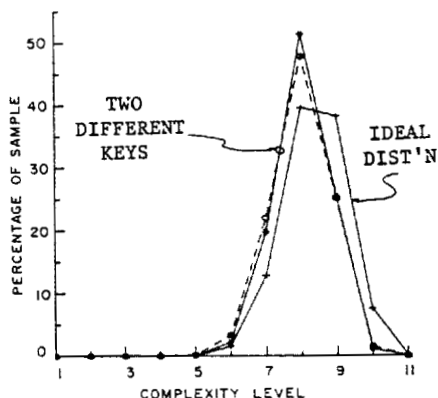


Fig. 2: Plaintext/ciphertext distribution of complexity for 32-bit modulo multiplication, modulo  $2^{32}$ . It can be seen that the distributions fall short of the ideal distribution.

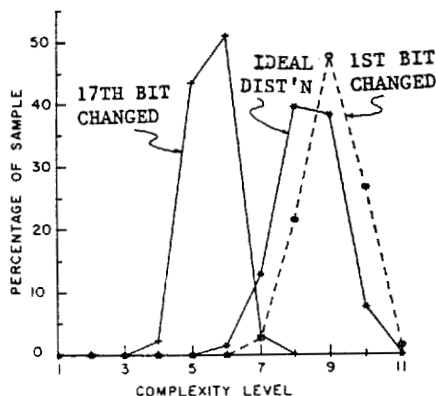


Fig. 4: Distribution of avalanche complexity for 32-bit multiplication, modulo  $2^{32}$ . The distribution depends on which plaintext bit is complemented to generate the avalanche effect.

could also be generated by selecting 64-bit blocks from a Binary Memoryless Source (BMS) with equiprobable symbols. Lempel and Ziv [1] showed that the distribution of complexity is related to the entropy of the source generating the sequences and this is illustrated by the other two curves in Fig. 1. The curve labelled BMS is generated by a Binary Memoryless Source with  $p(0) = 0.7$  and the curve labelled BSMS is generated by a Binary Symmetric Markov Source with  $p(0/0) = p(1/1) = 0.7$ . These two information sources have the same entropy (.881 bits/symbol) but different structure and it is seen that they are quite close together, but distinct from the ideal distribution. The threshold of complexity is given by  $C_n = 64/\log_2 64 = 10 \frac{2}{3}$ .

In an ideal block cryptographic system the plaintext vector  $\underline{P}$  and the corresponding ciphertext  $\underline{C}$  should appear to be independent of each other. Let

$$\underline{S} = \underline{P} \oplus \underline{C}$$

where  $\oplus$  means the exclusive OR sum of the two binary  $n$ -vectors, term by term. Then, for a well designed cryptographic algorithm,

$$C(\underline{S}) \geq n/\log n$$

with high probability, where  $C(\underline{S})$  is the complexity of the sequence  $\underline{S}$  (of length  $n$ ). If we pick a large number of plaintext sequences  $\underline{P}$  at random and compute  $C(\underline{S})$  in each instance, then the distribution of  $C(\underline{S})$  should appear as indicated by the 'IDEAL' curve in Fig. 1, where  $n = 64$  in this instance.

The complexity  $C(\underline{S})$  defined above may be referred to as plaintext/ciphertext complexity, since  $\underline{S}$  is the difference of  $\underline{P}$  and  $\underline{C}$ . In a similar manner, we can define a ciphertext/ciphertext or 'avalanche' complexity as follows. Let the plaintext vector  $\underline{P}$  generate the ciphertext  $\underline{C}$ , and  $\underline{P}'$  generate  $\underline{C}'$  where  $\underline{P}'$  is obtained from  $\underline{P}$  by complementing a bit in a designated location. Determine the  $n$ -vector

$$\underline{U} = \underline{C} \oplus \underline{C}'$$

where  $\underline{U}$  is a measure of the difference between the ciphertexts and thus is also a measure of the avalanche effect. In an ideal cryptographic algorithm,  $\underline{C}$  and  $\underline{C}'$  should appear to be statistically independent and thus  $\underline{U}$  should appear to be randomly selected from the set of all binary  $n$ -tuples. Letting  $C(\underline{U})$  represent the complexity of  $\underline{U}$ , it should also be true that

$$C(\underline{U}) \geq n/\log n$$

with very high probability. The distribution of avalanche complexity for a specified plaintext bit position can be estimated by selecting plaintext vectors at random and complementing the designated bit. The complexity of  $\underline{U}$ ,  $C(\underline{U})$ , is determined in each case. If the cryptographic algorithm is well designed, the distribution generated in this way should match very closely with the ideal distribution generated by the set of all binary vectors of length  $n$ . Note that the avalanche complexity distribution may be a function of the bit location that is complemented. Such variations would reveal cryptographic weaknesses. Avalanche complexity can also be defined by complementing key bits instead of plaintext bits. It should also be noted that the avalanche effect can be generalized by complementing a specified combination of bit positions.

#### THE COMPLEXITY TEST APPLIED TO MODULO MULTIPLICATION

The operations  $A*B \bmod 2^n$  and  $A*B \bmod 2^n-1$ , where  $A$  and  $B$  are binary  $n$ -vectors, are helpful for illustrating the complexity test. The operation  $*$  between  $A$  and  $B$  is binary multiplication, and reduction  $\bmod 2^n$  is easily implemented since overflow high order digits fall off the end. However, due to the fact that the carries propagate from right to left and the overflow drops off the end, the mixing effect is not uniform. To examine this more closely, we applied the complexity test to the operation  $A*B \bmod 2^n$ , for  $n = 32$ . One of the parameters, say  $B$ , is kept fixed and may be regarded as the key ( $B$  must be an odd integer for invertibility). The other,  $A$ , is a random 32-bit binary vector which is selected many times. The plaintext/ciphertext complexity test is performed for each choice of  $A$  and a distribution of complexity is obtained. This is shown in Fig. 2 and gives the average complexity averaged over the 32 bits. To exhibit the non-uniformity, we can perform the complexity test on the left half (high-order 16 bits) and right half (low-order 16 bits) separately. It can be seen from Fig. 3 that for the same choice of  $B$  (the "key"), the left half is more complex than the right half.

The avalanche complexity test was also performed for the operations  $A*B \bmod 2^n$  and  $A*B \bmod 2^n-1$ . It can be seen from Fig. 4 that the complexity distribution for the modulus  $2^n$  differs quite substantially from the ideal distribution, but the fit is much better for the modulus  $2^n-1$ . This can be seen by comparing Fig. 4 and Fig. 5, where Fig. 5 gives the avalanche complexity for modulus  $2^{32}-1$ . After

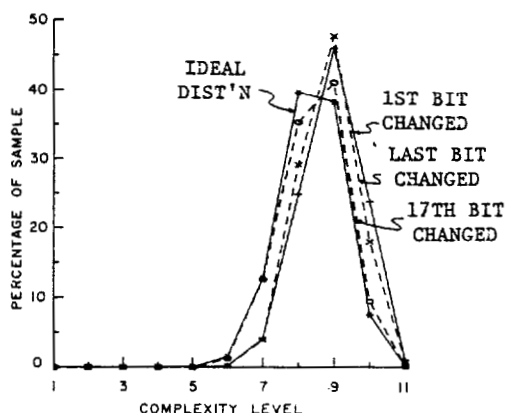


Fig. 5: Distributions of avalanche complexity for 32-bit modulo multiplication, modulo  $2^{32}-1$ . The curves are much closer to the ideal distribution than for modulo  $2^{32}$ .

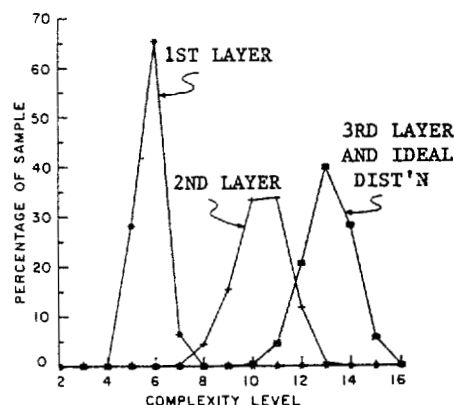


Fig. 7: Avalanche complexity for successive layers of DES produced by complementing the 32nd bit of plaintext. The curves for four or more layers are very close to the ideal distribution.

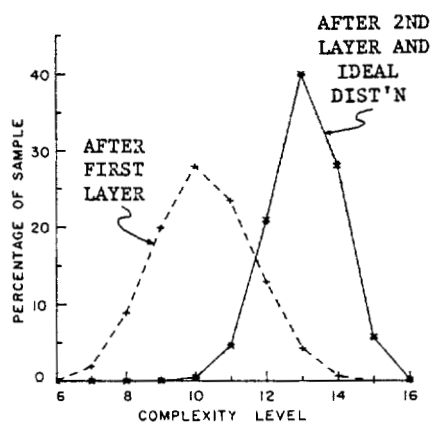


Fig. 6: Plaintext/ciphertext complexity for successive rounds (layers) of DES. From the 2nd layer on, the curves are indistinguishable from the ideal distribution.

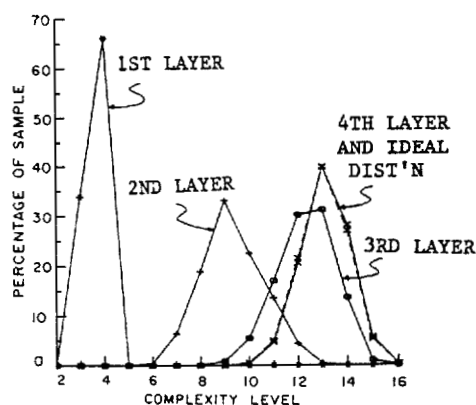


Fig. 8: Avalanche complexity for successive layers of DES produced by complementing the first bit of key. The curves are very close to the ideal distribution by the fourth layer.

a little reflection this should not be too surprising. For operations mod  $2^n - 1$ , the carries propagate around the end cyclically and the effect of the carry is much more uniform.

#### APPLYING THE COMPLEXITY TEST TO DES

It would be expected that the DES encryption algorithm should do well under the complexity tests, and this was found to be the case. What is also of interest is to observe how rapidly the DES algorithm approaches the ideal complexity distribution as we include more of the 16 rounds or iterations. (The initial and final permutations are ignored.) As shown in Fig. 6, the plaintext/ciphertext complexity converges to the ideal after the second iteration. However, the avalanche complexity requires four to five iterations before there is a good fit. This indicates that the avalanche complexity test is more demanding than the plaintext/ciphertext complexity test. The avalanche complexity test is performed by complementing a plaintext bit and a key bit as shown in Figs. 7 and 8, respectively.

#### ACKNOWLEDGEMENT

The authors wish to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada in the form of a Strategic Grant on Communications, Grant No. G0666.

#### REFERENCES

- [1] A. Lempel and J. Ziv, "On the complexity of finite sequences", IEEE Trans. on Inform. Theory, Vol. IT-22, Jan. 1976, pp. 75-81.
- [2] E. Fischer, "A Theoretical Measure of Cryptographic Performance", Cryptologia, Vol. 5, Jan. 1981, pp. 59-62.
- [3] E. Fischer, "Measuring cryptographic performance with production processes", Cryptologia, Vol. 5, Jul. 1981, pp. 158-162.
- [4] M.E. Spencer and S.E. Tavares, "Layered Broadcast cryptographic systems", Advances in Cryptology: Proceedings of Crypto '83, Plenum Press 1984, pp. 157-170.



# AN UPDATE ON QUANTUM CRYPTOGRAPHY

Charles H. Bennett<sup>1</sup> & Gilles Brassard<sup>2</sup>

<sup>1</sup>IBM Research, Yorktown Heights, NY 10598  
(Current address: Boston University  
Computer Science Department, Boston, MA)

<sup>2</sup>Université de Montréal, Département IRO,  
C.P. 6128, Succ. "A", Montréal (Québec), Canada H3C 3J7  
(Current address: University of California,  
Computer Science Division, 573 Evans Hall,  
Berkeley, CA 94720)

## INTRODUCTION

Although written about fifteen years ago, Wiesner's seminal paper, to which the origin of quantum cryptography must be traced back, did not appear in print until the spring of 1983 [W83]. The first published account of these ideas thus appeared in the proceedings of the second annual CRYPTO conference [BBBW83]. However, the concepts presented there were mostly of theoretical interest, because the technology involved in implementing them would have been far beyond the reach of our current knowledge. In particular, single polarized photons had to be trapped, bouncing back and forth between perfectly reflecting mirrors, and perfect efficiency in photon detection was required. To make up for this inconvenience, we could prove that no technology whatsoever, as well as no amount of computing power, could break some of our schemes, as long as some of the most fundamental principles of quantum physics hold true.

During the two years that have elapsed since, quantum cryptography has come a long way towards practicality. The most important breakthrough was quite an obvious observation: God did not create photons as a storage medium, but rather as a communications device. This paved the way to a quantum channel on which passive eavesdropping is meaningless, whereas any significant amount of active tampering has a high probability of being detected. The purpose of this Update is to present a short summary of the new results, and to stress how they differ from the current trend in cryptography.

## THE CURRENT TREND IN CRYPTOGRAPHY

Conventional cryptosystems, such as Enigma [G79], DES [NBS77] and even RSA [RSA78] are based on a mixture of mathematics, guesswork and wishful thinking. Shannon's information theory [Shan48, Shan49] does not take into account the amount of computing power at the enemy's disposal. On the other hand, the theory of computational complexity is not yet well enough understood to prove the computational security of public-key cryptosystems [DH76]. Even the theory of NP-completeness [GJ79] is unlikely to bear any relevance to cryptography [Br79].

The need for such proofs was dramatically emphasized when Shamir [Sham82, BS83] first explained at CRYPTO 82 how to break the basic Merkle-Hellman knapsack scheme [MH78]. Unfortunately, until the  $P=?NP$  question is settled [GJ79], the security of any public-key cryptosystem is doomed to depend on experience and unproved conjectures. The following quote from the original paper on (now broken) knapsack schemes is quite eloquent: "Faith in the security of these systems must therefore rest on intuition and on the failure of concerted attempts to break them" [MH78]. This is so reminiscent of what used to be said about World War II and earlier ciphers that one can only shiver at the thought that such is still the current situation. The following quotes, from an excellent tutorial introduction to cryptography by Diffie and Hellman, are certainly not obsolete, although some progress has been achieved in the past five years: "Cryptography is currently an engineering subject in which there are more facts and rules of thumb than theorems or systematic developments", and "We expect that provably secure systems will be developed as computer science progresses, but until that time, the current process of certification by mock attack will remain the most reliable test of a system's strength" [DH79].

Even the truly remarkable notion of probabilistic encryption, as set forth by Goldwasser and Micali in recent years [GM84], is not immune to an eventual breakthrough in algorithm design. The superb mathematics underlying these schemes can only serve to weaken the assumptions needed to infer their security. Nonetheless, they are also ultimately based on unproved conjectures in computational number theory. They have only changed the process of certification, which can concentrate on finding efficient algorithms for the relevant number theory problems, instead of working directly on pieces of ciphertext. Perhaps even more disturbing is the thought that such efficient algorithms may very well have been discovered already, but that they are being kept secret for obvious intelligence reasons, or in the hope of reaping a substantial profit.

It is nonetheless possible to prove *negative* theorems about mathematically based cryptosystems. For instance, Shannon proved that no traditional secret-key cryptosystem can achieve perfect secrecy against unlimited computing power, unless the key, used once only, is at least as long as the cleartext. Similarly, it is not hard to prove that any public-key distribution scheme [DH76] can be broken, given sufficient computing power, even if the cryptanalyst is only allowed passive eavesdropping.

## QUANTUM CRYPTOGRAPHY

The purpose of quantum cryptography is to propose a radically different foundation for cryptography, viz. the uncertainty principle of quantum physics [Bo51]. Quantum cryptography can achieve most of the benefits of public-key cryptography, with the additional advantage of being provably secure, even against an opponent with superior technology and unlimited computing power, barring fundamental violations of accepted physical laws. It can be roundly asserted that any successful attack on some of our schemes would have more far reaching consequences on contemporary physics than an efficient factoring algorithm, or even a proof that  $P=NP$  (sic), would have on mathematics and computer science. Perhaps even more remarkable is the fact that quantum cryptography allows for protocols that achieve both mathematically impossible feats discussed at the end of the previous section.

Offsetting these advantages is the practical disadvantage that quantum transmissions are necessarily very weak and cannot be amplified in transit. However, a recent experiment conducted in France by Aspect, Grangier and Roger [AGR82] in order to test the Einstein-Podolsky-Rosen-Bohm gedankenexperiment [EPR35, M81] clearly indicated that quantum cryptography is within the reach of current technology, although more work is necessary for it to become economical and practical. Another disadvantage of quantum cryptography is that it does not provide digital signature [DH76] and related features, such as certified mail [Bl83a] or the ability to settle a dispute before the judge. However, these limitations seem to be inherent to any scheme secure against unlimited computing power. Also, the proposed coin tossing scheme discussed below is not secure against very advanced technology.

Readers interested in implementation details of the various quantum cryptography schemes are referred to other conference proceedings [BBBW83, BB83, BB84]. Let us only briefly describe here the basic underlying

principles. In conventional information theory and cryptography, it is taken for granted that digital communications can always be monitored and copied, even by someone ignorant of their meaning. Such copies can be stored for an eventual future use, such as helping the decryption of later transmissions enciphered with the same secret key. However, when elementary quantum systems, such as polarized photons, are used to transmit digital information, the uncertainty principle gives rise to novel cryptographic phenomena, unachievable with traditional transmission media. This principle can be used effectively to design a communications channel whose transmissions in principle cannot be read or copied reliably by an eavesdropper ignorant of certain key information used in forming the transmission. The eavesdropper cannot even gain partial information about such a transmission without altering it in a random and uncontrollable way, likely to be detected by the channel's legitimate users.

Such a channel allows the unlimited re-use of a one-time pad without any breach of security, thus contradicting a well-established theorem of Shannon's. Whenever eavesdropping occurs, the enemy can gain no information on the message that was being sent, but the channel's legitimate users are warned that eavesdropping was attempted. A new secret key must then be used to retransmit the previous message, as well as for all further transmissions. As this new key could have been sent through the quantum channel as a previous secure transmission using an older key, this scheme has been described as a self-winding one-time pad.

More interestingly, the quantum channel achieves one of the main advantages of public-key cryptography by permitting secure distribution of random key information between two parties who share no secret information initially, provided both parties have access, beside the quantum channel, to an ordinary channel susceptible to passive eavesdropping, but not to active tampering. Even in the presence of active tampering, the two parties can still distribute a key securely if they share some much shorter secret information initially, provided the tampering is not so frequent as to suppress communications completely. These key distribution and key expansion schemes remain secure even if the enemy has unlimited computing power. Recall that it is a theorem that this is impossible to achieve for mathematically based schemes.

Finally, we also have a protocol for coin tossing [B183b] by exchange of quantum messages, which is secure against traditional kinds of cheating, even by an opponent with unlimited computing power. Ironically, it can be subverted by use of a still subtler quantum phenomenon,

the already mentioned Einstein-Podolsky-Rosen-Bohm gedankenexperiment. This threat is merely theoretical, however, because it requires perfect efficiency of storage and detection of photons, which though not impossible in principle, is far beyond the capabilities of current technology. The honestly followed protocol, on the other hand, could be realized with current technology.

There is an interesting similarity between probabilistic encryption and quantum cryptography: both rely on the notion of reduction. However, whereas the former reduces the unproved computational complexity of some outstanding problems of number theory to the difficulty of breaking the schemes, the latter relies on the most fundamental beliefs of quantum physics. For instance, one such reduction can be used to prove about one of the coin tossing opponents that any systematic advantage he could get on the outcome of the coin toss could be used to effectively transmit information faster than the speed of light.

#### REFERENCES

- [AGR82] Aspect, A., P. Grangier and G. Roger, "Experimental Realization of the Einstein-Podolsky-Rosen-Bohm Gedankenexperiment: a New Violation of Bell's Inequalities", *Physical Review Letters*, Vol. 49, pp. 91-94 (1982).
- [BB83] Bennett, C. H. and G. Brassard, "Quantum Cryptography, and its Application to Provably Secure Key Expansion, Public-Key Distribution, and Coin Tossing", *IEEE International Symposium on Information Theory*, St-Jovite, Quebec (1983).
- [BB84] Bennett, C. H. and G. Brassard, "Quantum Cryptography: Public-Key Distribution and Coin Tossing", *Proceedings of the International Conference on Computers, Systems and Signal Processing*, Bangalore, India (1984).
- [BBBW83] Bennett, C. H., G. Brassard, S. Breidbart and S. Wiesner, "Quantum Cryptography, or Unforgeable Subway Tokens", *Advances in Cryptography: Proceedings of CRYPTO 82*, Plenum Press, pp. 267-275 (1983).
- [Bl83a] Blum, M., "How to Exchange (Secret) Keys", *ACM Transactions on Computer Systems*, Vol. 1, no 2, pp. 175-193 (1983).
- [Bl83b] Blum, M., "Coin Flipping by Telephone - A Protocol for Solving Impossible Problems", *SIGACT NEWS*, Vol.15, no 1, pp. 23-27 (1983).
- [Bo51] Bohm, D., *Quantum Theory*, Prentice Hall, Englewood Cliffs, NJ (1951).
- [Br79] Brassard, G., "A Note on the Complexity of Cryptography", *IEEE Transactions on Information Theory*, Vol. IT-25, no 2, pp. 232-233 (1979).
- [BS83] Brickell, E. F. and G. J. Simmons, "A Status Report on Knapsack Based Public Key Cryptosystems", *Congressus Numerantium*, Vol. 37, pp. 3-72 (1983).

- [DH76] Diffie, W. and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, Vol. IT-22, pp. 644-654 (1976).
- [DH79] Diffie, W. and M. E. Hellman, "Privacy and Authentication: an Introduction to Cryptography", *Proceedings of the IEEE*, Vol. 27, no 3, pp. 397-427 (1979).
- [EPR35] Einstein, A., B. Podolsky and N. Rosen, *Physical Review*, Vol. 47, p. 777 (1935).
- [G79] Garlinski, J., *The Enigma War*, Charles Scribner's Sons, New York, NY (1979).
- [GJ79] Garey, M. R. and D. S. Johnson, *Computers and Intractability, a Guide to NP-Completeness*, W. H. Freeman and Co., San Francisco, CA (1979).
- [GM84] Goldwasser, S. and S. Micali, "Probabilistic Encryption", *Journal of Computer and System Sciences*, Vol. 28, pp. 270-299 (1984).
- [M81] Mermin, N. D., "Bringing Home the Atomic World: Quantum Mysteries for Anybody", *American Journal of Physics*, Vol. 49, no 10, pp. 940-943 (1981).
- [MH78] Merkle, R. C. and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks", *IEEE Transactions on Information Theory*, Vol. IT-24, no 5, pp. 525-530 (1978).
- [NBS77] ---, "Data Encryption Standard", *National Bureau of Standards*, FIPS PUB 46, Washington, DC (1977).
- [RSA78] Rivest, R. L., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, Vol. 21, no 2, pp. 120-126 (1978).
- [Sham82] Shamir, A., "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem", *Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science*, pp. 145-152 (1982).
- [Shan48] Shannon, C. E., "A Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, pp. 379-423, 623-656 (1948).
- [Shan49] Shannon, C. E., "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, Vol. 28, pp. 656-715 (1949).
- [W83] Wiesner, S., "Conjugate Coding", unpublished manuscript written ca 1970, subsequently made available in *SIGACT NEWS*, Vol. 15, no 1, pp. 78-88 (1983).

HOW TO KEEP A SECRET ALIVE  
*EXTENSIBLE PARTIAL KEY, KEY SAFEGUARDING, AND THRESHOLD SYSTEMS*

David Chaum

Center for Mathematics and Computer Science (CWI)  
Kruislaan 413  
1098 SJ Amsterdam, The Netherlands

PREVIOUS WORK

Partial key, key safeguarding, and threshold techniques appear to be another example of similar good ideas springing up in several places at nearly the same time—each with a different name and associated terminology. The use of partial key techniques actually appeared in print first in a technical report [Chaum 79] before the key safeguarding techniques were presented at a conference [Blakley 79], and before the threshold schemes were submitted for publication [Shamir 79]. (In fact the author received comments on the technical report from Shamir, along with a draft of the threshold scheme.)

The essential idea of all three techniques is that someone who knows a secret number can form other numbers from it, such that it is easy to compute the secret number if you know any fixed  $k$  of the other numbers, but knowing less than  $k$  of the other numbers gives no clue about the secret number.

Feistel proposed dividing a key into parts such that the key could be recovered by forming the bitwise exclusive-or of all the parts [Feistel 70]. Shamir quotes a problem from a combinatorics text in which all fixed-size subsets of a set of scientists have sufficient physical keys to unlock a file cabinet protected by multiple padlocks [Liu 68]. The partial key technique, which works with cryptographic keys, was an improvement over the approach of the padlocks, because it allowed each trustee of keys to hold only one key, instead of many keys. It might be quite convenient in practice for small numbers of trustees, and can provide unconditional security. It appeared as a single paragraph and footnote in a proposal for distributed computer systems that can be trusted by groups who don't necessarily trust each other. The secret sharing and threshold techniques are far more elegant than the original partial key technique. They allow large systems in practice, and also can provide unconditional security. A number of similar

schemes have appeared subsequently.

Since no standard terminology seems to have emerged, the following will be used: in a partial key system, the *system creator* divides the *key* into *partial keys* (Blakley's shadows, Shamir's pieces) that are transmitted to various *trustees* (Blakley's guards), such that any *quorum* (Shamir's threshold) of trustees is sufficient to *recover* the key, and less than a quorum of trustees is insufficient.

The present work describes techniques allowing any partial key system to adapt for survival in the face of changing availability of trustees and even changing needs for system parameters.

## MOTIVATION

All trustees may not always remain able and willing to recover the key in a partial key system. For example, a computer acting as a trustee and storing a partial key may be destroyed by natural or other disaster, or a human trustee may be hit by a truck. Less severe causes can also easily be imagined, such as hardware or software failures, or a person suffering from loss of memory. A trustee that will never participate in recovering a key will be said to be *lost*; a trustee able and willing to participate will be said to be *present*.

Clearly it is prudent to consider scenarios in which trustees become lost. If less than a quorum of trustees remain after one or more trustees is lost, then it has become impossible to recover the key. Problems could also result from a substantial loss that leaves only a quorum of trustees, or relatively few more than a quorum, for reasons similar to those for having the greater number of trustees in the original system. For example, a loss of sufficient trustees to prevent recovery might then become too likely or too easy to cause, or some cooperating trustees might gain significant power from their ability to prevent recovery of the key.

A solution to the problems of loss of trustees is presented in the next section. The section after that considers causes of unavailability of trustees other than simple loss.

## REPLACING A TRUSTEE

The problem of loss of trustees is solved by allowing new trustees to *replace* lost trustees. If some trustee is lost, and a quorum of trustees is present, then the quorum can give a replacement trustee the same ability that the lost trustee had to participate in the partial key system; the replacing trustee is given the slot of the lost trustee replaced.

Any partial key technique can be used in a way that allows such replacement. The ability to allow replacement can be "built-in" when the partial key system is first created, or it can be "added-on" later separately by each trustee. The added-on approach is considered first for a single trustee and next for all trustees. Then the built-in approach is considered.

Suppose you as a trustee wish to make provisions that would allow your own replacement,



should it ever become necessary. The essential idea of the solution is that you create your own partial key system to allow the other trustees to recover your partial key. You divide your partial key into *sub-partial* keys. The quorum parameter you use is the same as that in the original partial key system, and the number of partials you generate is one less than the number of original trustees. Then you transmit a different sub-partial to each other trustee. (You might use cryptographic techniques, e.g., to provide secrecy and authentication of sub-partials transmitted.) If you should become lost, heaven forbid, and some present quorum wants to replace you by filling your slot with a replacing trustee, then each member of the quorum would separately transmit to the replacing trustee the sub-partial they received from you. (Again these transmissions might be cryptographically protected.) When the replacement gets a quorum of your sub-partials, the replacement is able to recover your partial—and has become able to participate in the original system in your place.

If other trustees try to provide for their own replaceability in a similar way, problems may eventually arise since replacements won't have access to sub-partials, and consequently won't be able to help make subsequent replacements. Consider a solution in the homogeneous case of a set of trustees who each provide every other trustee with sub-partials. After a trustee receives sub-partials from all the other trustees, the trustee encrypts them together using the trustee's partial key as a key in say a conventional cryptosystem. This collection of all the sub-partials received by a trustee, encrypted by the trustee's partial key, will be called an *extender*. Thus, once a replacement gets sufficient sub-partials to allow recovery of the partial, the replacement can use the partial to decrypt the extender and obtain the full collection of sub-partials that was available to the replaced trustee.

Unlike the key and partials, extenders are public. But to be of use they must of course be accessible. One way to treat extenders is to allow them to be copied freely, and assume that this provides adequate protection against all copies becoming inaccessible. Another way is for trustees to keep copies of extenders.

The built-in approach promised above is easily seen by noticing that the original system creator knows all the partials and can thus form sub-partials, and from them the extenders. The system creator might, for example, transmit the sub-partials to the trustees along with the partials, and make the extenders public.

## ADDING A NEW TRUSTEE

It might be desired to add new trustees without replacing any specific trustee. One reason might be just to expand the reliability of the system. Another reason is to be able to contend with *missing* trustees, i.e. those that are not present and not known to be lost. For example, a trustee might not be reachable because of a communication failure or other circumstances, or whether a trustee will recover from some disabled state may not be known with certainty. A kind of reversible replacement, even if such were possible, may not be the best approach. Consider for example the case where a missing trustee returns just making a quorum, but

recovery is impossible since the replacing and returned trustees together can contribute only one partial. Thus it may be desirable to compensate for missing trustees by *adding* new trustees.

Now a technique for allowing new trustees to be added is presented. It is a built-in approach, requiring the system creator to make provisions for the additions when the system is created. The total number of trustees that can be added must be fixed before the system is created (but see the next two sections).

The technique is essentially the same as the built-in approach of the previous section, except that some partials are created that are not issued to trustees initially, and extenders need only cover these partials. The system creator provides each initial trustee with the usual partial and also with a different sub-partial for each trustee slot that can be added. A quorum of the sub-partial for a particular such slot allows recovery of the partial for that slot. When the sub-partial is transmitted to a new trustee, the new trustee uses them to recover the appropriate partial. This partial allows the new trustee to participate in recovering the original key—just as any other trustee. Systems providing for the addition of more than one new trustee could use extenders to ensure that new trustees are themselves as capable as the other trustees of allowing new trustees to be added. Thus, it would be sufficient for the system creator to form an extender for each new trustee slot, such that each extender contains sub-partial for all the other new trustee slots. If the order in which new slots will be filled is fixed when the system is created, then the extender for a particular new slot need only contain sub-partial for the new slots that appear after it in the order. (A similar technique is buried in [Chaum 82].)

## REPLACING A SYSTEM

It might be desired to replace a partial key system that is in use by a new partial key system—without using a mutually trusted party like the original system creator. There are several reasons for wanting to do this: to change the parameters of the existing system, such as the quorum size and the number of trustees; to change the set of trustees; or to restore the built-in ability to add trustees if the original provisions become depleted. Of course the extent of effective change may be limited because some trustees may not be relied on to destroy their old partial keys. If the number of trustees not destroying old partials is less than the old quorum, then the situation is effectively the same as if all had destroyed their partials. A quorum of trustees acting together will be able to *replace* an existing system with a new system.

The essential idea is the same as with replacing or adding a trustee, except that the parameters and trustees of the sub-partial may differ from those of the replaced system. What is in effect created is an old quorum of partial key systems, each of whose parameters and trustees are the same, and each of whose keys is a partial of the replaced system. Thus, a new partial is actually a *collection* of partials, one from each of the old quorum who established the new system. A new quorum of such collections is sufficient to recover an old quorum of old partials, and thus the original key. Extenders are used only to provide for adding trustees in the new system, with each trustee issuing them as the system creator did in the previous section. Of course it should

be ensured that old partials will not be destroyed until the new system is in place.

If a newer system is to replace a new system, then each collection of the new system would be encrypted under a key created by the trustee holding the collection, and then made public or otherwise protected like extenders. The key created by each trustee would be used as the key in a partial key system, with the newer parameters, whose partials would be transmitted to the newer trustees. (A similar technique is also buried in [Chaum 82].) For arbitrary fixed maximum system size and parameters, the time and space requirements of a series of systems succeeding each other in this way are linear in the length of the series.

## OPEN QUESTIONS

More natural and efficient mechanisms for extensibility seem possible.

## CONCLUSIONS

Various ways to allow the trustees of a key to adjust to changes in their own membership and external circumstances have been described. The techniques appear to be quite flexible and potentially quite useful in practice.

## REFERENCES

- (1) Blakley, G.R., "Safeguarding Cryptographic Keys," Proceedings A.F.I.P.S. 1979 National Computer Conference, vol. 48, June 1979, pp. 313-317.
- (2) Chaum, D., "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups," Memorandum No. UCB/ERL M79/10, University of California, Berkeley, CA, February 22, 1979.
- (3) Chaum, D., "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups," dissertation, Computer Science, University of California, Berkeley, CA, June 1982.
- (4) Feistel, H., "Cryptographic Coding for Data-Bank Privacy," Research Report RC 2827, IBM T. J. Watson Research Center, Yorktown Heights, NY, March 1970.
- (5) Liu, C.L., Introduction to Combinatorial Mathematics, McGraw Hill, NY, 1968.
- (6) Shamir, A., "How to Share a Secret," Communications of the ACM, vol. 22, no. 11, November 1979, pp. 612-613.

## Author Index

- Akl, Selim G. 269  
Beker, Henry 401  
Bennett, Charles H. 475  
Blake, I.F. 73  
Blakley, G.R. 242, 314  
Blum, Manuel 289  
Brassard, Gilles 475  
Brickell, Ernest F. 342  
Brookson, C.B. 3  
Chaum, David 432, 481  
Chen, Su-shing 95  
Chor, Benny 54, 303  
Clark, B.L. 3  
Coulthart, K.B. 203  
Davies, Donald Watts 393  
Davio, Marc 144, 359  
Davis, J.A. 114  
Desmedt, Yvo 144, 147, 359  
El Gamal, Taher 10  
Fairfield, R.C. 115, 203  
Fortune, Steven 454  
Goubert, Jo 144, 147  
Goldreich, Oded 276, 303  
Goldwasser, Shafi 276, 289, 467  
Holdridge, D.B. 114  
Hoornaert, Frank 144, 147  
Kaliski, Burt S. 83  
Konheim, Alan G. 339  
Kothari, S.C. 231  
Leighton, Albert C. 101  
Leung, A.K. 468  
Magyarik, Marianne R. 19  
Manferdelli, J.L. 377  
Matusevich, A. 115  
Matyas, Stephen M. 101  
Meadows, Catherine 242  
Meijer, Henk 269  
Merritt, Michael 454  
Micali, Silvio 276, 467  
Mortenson, R.L. 203  
Mullin, R.C. 73  
Ong, H. 37  
Plany, J. 115  
Proctor, Norman 174  
Quisquater, Jean-Jacques 144, 359  
Reeds, J.A. 377  
Rivest, Ronald L. 54, 467  
Shamir, A. 37, 47  
Schnorr, C.P. 37  
Serpell, S.C. 3  
Simmons, Gustavus J. 411  
Tavares, S.E. 468  
Tedrick, Tom 434  
Vanstone, S.A. 73  
Vazirani, Umesh V. 193  
Vazirani, Vijay V. 193  
Wagner, Neal R. 19  
Walker, Michael 401  
Williams, H.C. 66  
Yung, Mordechai 439

## Keyword Index

- ADFGVX 339
- Affine 389
- Algebraic integers 41, 334
- Analog encryption 83
- Analog signal 96
- Associativity 321, 329
- Authentication 209, 413, 416, 428
  - channel 419, 435
  - system 417
  - theory 413
- Authenticators 393, 396
- Avalanche 359
- Avalanche complexity 468
- Bandwidth 84
- Bankers Automated Clearing Service 393
- Barbeau-Dubourg 105
- Beale ciphers 107
- Blind signatures 432
- Book cipher 101, 103
- Bose-Chowla theorem 54, 60
- Caesar cipher 321
- Card key 404
- Cascaded cipher system 174
- Channel capacity 421, 422, 427
- Chosen cyphertext attack 298
- Cipher feedback system 174
- Closure 329
- Coding theory 410, 412
- Coin tossing 477
- Collusion 459
- Complexity based cryptography 440
- Complexity of routing 147
- Concealing set 245
- Confusion 314, 315
- Contract signing 434, 438
- Coset 322, 323
- Credentials 432
- Cryptanalysis of knapsack cryptosystems 342
- Cryptosystem factorization 378
- Data encryption algorithm 403
- Data seal 393
- DEP 115
- DES 115, 144, 147, 269, 359, 377, 468
- Dictionary 101
- Diffusion 314, 315
- Digital
  - encryption processor 115
  - pseudonyms 432
  - signatures 10, 12, 13, 15, 37
- Discrete logarithms
  - 10, 12, 14, 15, 17, 55, 57, 59, 60, 63, 73
- Dumas, Charles W.F. 106
- Electronic funds transfer 401
- Electronic signature 401
- Equivalent to factoring 467
- Error propagation 174
- Euclidean algorithm 75
- Extensible partial key systems 481
- Exchange of secret keys 434, 436, 438
- Factor 377
- Factorization
  - 15, 17, 21, 66, 114, 193, 194, 291, 303, 441
- Fractionation 339
- Fractions of a bit 434-437, 438

- Franklin, Benjamin 105
- Game theory 413, 417
- General position 235
- $GF(2^{127})$  74
- Group 22
- Group-theoretic formulation 322
- Guard 481
- Haldimand 106
- Hard as factoring 66, 194
- Hardware 118, 144, 147
- Hashing function 277, 281
- Hierarchical threshold scheme 239, 240
- Identity based cryptosystems 47
- Identify friend or foe system 283
- Index calculus 74
- Inversion secure 197
- ISO 393
- Jacobi symbol 67
- Key
  - book 105
  - clustering 359
  - expansion 478
  - management 401
  - register 404
  - safeguarding 481
- Kluber 107
- Knapsack cryptosystems 54, 55, 60, 341
- Las Vegas algorithm 197
- Lattice basis reduction 343, 344
- Lie algebra 95, 97
- Lie group 95, 97, 99
- Limitation of pin connections 147
- Linear congruence 269, 271
- Linear shift register 269
- Linear threshold schemes 232, 244
- Linear variety 233
- Low density attacks 54, 60
- LSI 115, 203
- Maximization of speed 147
- Mental poker 439
- Message authenticator algorithm 393
- Message position 320
- Minimization of chip area 147
- Mixing 329
- Modified RSA scheme 66
- Modulo multiplication 468
- Multi-player mental poker 439
- NP-complete problems 19, 20
- Number embedding 441
- Oblivious transfer 439
- One-parameter subgroup 95, 98
- One-time pad 459, 478
- One-way functions 269, 277, 278, 284
- Oscillator 205
- OSS signature scheme 51
- Partial key 481
- Partial information 290
- Passive adversaries 296
- Period 270, 271, 273
- Permutations 270, 271, 455
- PIN 401
- Plastic card 401
- Polyalphabet 372
- Polynomial equations 38
- Polynomial interpolation 441
- Probabilistic encryption 291
- Prolate spheroidal sequences 88
- Propagation 361
- Protocol 454
- Prototype encryption system 6
- Pseudonyms 432
- Pseudo-random 193, 194
  - function 277-279
  - bit generator 277-279, 283-286
  - number generators 194
  - sequences 269, 271, 272
- p/s/r process 242
- Public key 3-5
  - cryptosystem 10, 11, 15, 17, 19, 23, 440
  - distribution 10-12, 73, 477
  - encryption 434, 438
  - exponentiator 7
- Quadratic sieve 114
- Quantum cryptography 475

- Rabin 303
- Ramp scheme 243
- Random element 95, 98
- Random number generator 203
- Randomized encryption 24
- Ramp scheme 243
- Recover 481
- Replacement 314, 315
- Revealing map 245
- Risk seeking/Risk adverse 434, 438
- RNG 203
- Rotation group 95
- RSA 15, 16, 49, 193, 194, 303
- Running key 102
- Secret blocking 439
- Self-synchronization 176
- Semigroup 23
- Sequence complexity 468
- Shadow 244, 481
- Shannon perfectly secure 242
- Shannon relatively secure 244
- Signature problem 467
- Signature schemes 37, 50
- Speech 83
- Statistical test 278, 279
- Subscription to a key 439
- S.W.I.F.T. 393
- Systematic equations 76
- Terminal 402
- Threshold schemes 242, 481
  - rigid 245
  - generalized 231, 238
- Torsion 327
- Transaction key 404
- Trapdoor 19, 25, 27
- Trist, Nicholas 107
- Uncertainty principle 404, 478
- Undecidable problem 21, 22
- Unlimited computing power 478
- Untraceability 432
- Weight manipulation 77
- Word problem 22, 23
- $x^2 \bmod N$  generator 193, 194, 195
- XOR-Condition 193, 194